ANALISA ALGORITMA RIJNDAEL DAN TWOFISH

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat

Untuk Memperoleh Gelar Sarjana Teknik Pada

Jurusan Teknik Informatika

Oleh :

I S K A N D A R

10045017926

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU

PEKANBARU

2006

ANALISA ALGORITMA RIJNDAEL DAN TWOFISH

ISKANDAR

10045017926

Tanggal Sidang : 14 Juni 2006

Periode Wisuda : Februari 2007

Jurusan Teknik Informatika

Fakultas Sains dan Teknologi

Universitas Islam Negeri Sultan Syarif Kasim Riau

ABSTRAK

Rijndael adalah teknologi pengganti *Data Encryption Standard* (DES), dan Twofish merupakan algoritma kriptografi *block* yang merupakan salah satu kandidat sekaligus finalis *Advance Encryption Standard* (AES).Twofish mengadobsi jaringan Feistel yang merupakan teknologi jaringan yang digunakan pada DES, sedangkan Rijndael menggunakan Jaringan Substitusi dan permutasi berulang. Untuk mengetahui performa dan kehandalan kedua algoritma dari segi struktur, arsitektur, komponen *block* dan set instruksi yang digunakan pada kedua algoritma, untuk itu dilakukan dilakukan penelitian untuk menganalisa dan menguji kedua algoritma. Rijndael dengan jaringan substitusi dan permutasi yang berulang dengan operasi dominan seperti *SubByte*, S-BOX, *ShiftRow*, *MixColoumn* dan *AddRoundKey* (ARK) menghasilkan performa yang lebih baik dibandingkan Twofish yang mengadobsi jaringan Feistel dengan operasi-operasi seperti fungsi-F untuk memproduksi S-BOX, MDS, PHT, Rotasi *Bit* dan *whitening* (*Input Whitening* dan *Output Whitening*), dan sebaliknya untuk segi keamanan Twofish dengan lebih baik dibandingkan Rijndael.

Kata kunci : Kriptografi, Rijndael, Twofish.

ANALYSIS OF RIJNDAEL AND TWOFISH ALGORITHM

ISKANDAR

10045017926

Date of Final Exam : 16 June 2006

Graduation Cremony Priod : February, 2007

X Engineering Departement

Faculty of Sciences and Technology

State Islamic University of Sultan Syarif Kasim Riau

ABSTRACT

Rijndael is a replacement technology for the Data Encryption Standard (DES), and Twofish is a block cryptographic algorithm, which is one candidate and finalist for Advance Encryption Standard (AES). Twofish adopting a Feistel network is a network technology that is used in DES, Rijndael using Network Substitution and repeated permutations. To determine the performance and reliability of both algorithms in terms of structure, architecture, components and block instruction set used in the both algorithms, for it was done the research to analyze and test the algorithms. Rijndael with substitution-permutation Network operations are repeated with such dominant SubByte, S-BOX, ShiftRow, MixColoumn and AddRoundKey (ARK) resulted in better performance than that adopting Twofish Feistel network with operations such as function-F to produce the S-BOX , MDS, PHT, Bits Rotation and whitening (input whitening and output whitening), and for security aspect Twofish better than Rijndael.

Key : Cryptography, Rijndael, Twofish

KATA PENGANTAR

Assalammualaikum wr.wb

Puji syukur penulis ucapkan kehadirat Allah S.W.T yang telah memberikan limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul: "***ANALISA ALGORITMA RIJNDAEL DAN TWOFISH***".

Tugas akhir ini disusun sebagai salah satu persyaratan untuk menyelesaikan jenjang studi Strata-1(S-1) di Jurusan Teknik Informatika, Fakultas Sains Dan Teknologi, UIN SUSKA Riau.

Tujuan tugas akhir ini adalah menganalisa algoritma rijndael yang merupakan pengganti DES (*Data Encryption Standard*) dengan twofish sebagai salah satu kandidat dan juga finalis AES. Analisa kedua algoritma mencakup pada aspek kecepatan, tingkat kepadatan dan distribusi *bit* sehingga diharapkan dapat memberi informasi dan kemudahan mengenai kelebihan dan kekurangan dari masing-masing algoritma

Dalam kesempatan ini, penulis menyampaikan banyak terima kasih kepada pihak - pihak yang telah membantu penulis dalam penyusunan tugas akhir ini, yaitu kepada :

1. Bapak Prof. Dr. H. M. Nazir selaku Rektor Universitas Islam Negeri Sultan Syarif Kasim Riau.
2. Bapak Prof. Dr. H. Adrianto Ahmad, MT, selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Sultan Syarif Kasim Riau.
3. Ibu Fitri Wulandari, M.Kom, selaku ketua jurusan teknik Informatika.
4. Bapak Benny Sukma Negara, ST, selaku dosen pembimbing yang juga telah memberi bimbingan, arahan, dan saran yang berharga untuk tugas akhir ini.
5. Lestari Handayani, ST, selaku koordinator tugas akhir di teknik Informatika Universitas Islam Negeri Sultan Syarif Kasim Pekanbaru.

6. Bapak Novriyanto, ST selaku Penguji I dan bapak M. Irsyad, ST selaku Penguji II.
7. Dosen-dosen jurusan Teknik Informatika, yang selalu mengingatkan agar penulis cepat menyelesaikan tugas akhir.

Penulis menyadari bahwa penulisan tugas akhir ini belum sempurna dan masih banyak kekurangan. Oleh karena itu, kritik dan saran sangat diharapkan penulis untuk menuju perbaikan yang lebih baik. Semoga laporan tugas akhir ini bisa bermanfaat bagi siapapun yang membacanya.

Wassalamu'alaikum Wr. Wb

Pekanbaru, 14 Juni 2006

Penulis,

ISKANDAR

DAFTAR ISI

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kriptografi merupakan suatu teknologi yang digunakan untuk menjaga keutuhan dan keamanan informasi. Banyak teknologi kriptografi sekarang ini yang dikembangkan dan digunakan baik itu yang berupa *block* maupun *stream* pengguna teknologi ini.

DES (*Data Encryption Standard*) teknologi kriptografi yang telah digunakan sejak tahun 1977. Para kriptograper mengkhawatirkan teknologi ini mulai usang dan tidak aman digunakan, untuk mengantisipasi hal ini NIST (*National Institute of Standards and Technology*) Mengembangkan teknologi kriptografi standar baru sebagai pengganti dan kemudian terpilihlah Rijndael dan dinamakan AES (*Advanced Encryption Standard*).

Rijndael Merupakan teknologi kriptografi menggunakan teknologi SPN (*Substitution-Permutation Network). Substitution-Permutation Network* merupakan komponen dasar yang digunakan oleh teknologi kriptografi blok. Pada *network* ini terdapat *S-Box* ataupun *P-Box* dimana setiap *bit input* ditransformasikan ke *bit output*.

Twofish menggunakan Feistel *network,* teknologi ini merupakan teknologi sama yang digunakan *Data Encryption Standard*, dimana *bit input* dipecah menjadi beberapa bagaian dimana setiap bagiannya dilakukan operasi *S-Box*, *P-Box* maupun *Linear Mixing* menjadi beberapa bagian Feistel *network* adalah keduanya merupakan teknologi kriptografi kunci simetris karena menggunakan kunci yang sama pada saat enkripsi maupun dekripsi.

Tujuan penggunaan teknologi kriptografi adalah untuk pengamanan data/informasi tetapi implementasi pada bahasa pemrograman dan juga akselerasi teknologi pada implementasi *hardware* juga merupakan faktor penting yang perlu

diperhatikan, karena tidak semua operasi-operasi pada teknologi dapat berjalan baik pada sistem operasi, bahasa pemrograman dan hardware.

Pada penelitian ini pengujian dilakukan untuk mengetahui performa teknologi kedua algoritma Rijndael dan Twofish dengan input data 128 *bit* dan panjang key 256 *bit* dengan mode kriptografi standar yakni ECB, CBC, CFB dan OFB serta CTS yang merupakan varian dari CBC, analisa implementasi *library* Delphi Encryption Compedium menggunakan bahasa pemrograman Delphi.

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang, permasalahan yang diteliti yaitu bagaimana performa implementasi komponen, struktur, arsitektur *block* dan set instruksi yang digunakan. dalam analisis, pembahasan dan pengujian algoritma Rijndael dan Twofish menggunakan *bit input* (128 *bit*) dan kunci (256 *bit*) yang sama pada kedua algoritma dengan menggunakan alat penelitian berupa program uji yakni Delphi Encryption Compedium .

1.3 Batasan Masalah

Dalam Penelitian ini penulis memberikan beberapa batasan masalah pada penelitian ini, yakni:

1. Dilakukan analisa pada algoritma Rijndael dan Twofish (pada *block cipher* 128 dan panjang *key* 256)
2. Analisa dilakukan pada paket Delphi yakni Delphi Encryption Compedium selanjutnya di sebut DEC.
3. Pengujian dilakukan menggunakan data masukan *(input/plaintext*), *key*, Alat penelitian yang identik.

1.4 Tujuan Penelitian

Tujuan penelitian yang ingin dicapai dalam pelaksanaan tugas akhir ini adalah :

1. Mempelajari teknologi kriptografi data khususnya pada Twofish dan Rijndael.
2. Menganalisa algoritma Rijndael dan Twofish
3. Mengetahui konsep, arsitektur dan proses-proses enkripsi dan dekripsi khususnya pada kedua algoritma enkripsi tersebut.
4. Mengetahui kemungkinan kelemahan dan kekuatan masing-masing algoritma.

1.5 Metodologi Penelitian

Bab ini berisikan tentang tahapan penelitian yang meliputi tahapan pengumpulan data, analisa algoritma Rijndael dan Twofish, analisa dan perbandingan struktur, arsitektur, komponen *block*, set instruksi algoritma Rijndael dan Twofish, serta pengujian algoritma Rijndael dan Twofish menggunakan Delphi Encryption Compedium.

1.6 Sistematika Penulisan

Adapun sistematika penulisan akan dibahas menjadi 6 (enam) bab. Adapun rincian sebagai berikut :

BAB I Pendahuluan

Pada bab ini membahas latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, dan sistematika penulisan buku tugas akhir yang akan dibuat.

BAB II Landasan Teori

Pada bab ini mambahas teori–teori dasar kriptografi, matematika kriptografi, Struktur jaringan, arsitektur *block*, komponen *block*, set instruksi pada kedua algoritma dan Delphi Encryption Compedium sebagai media pengujian.

BAB III Metode Penelitian

Pada bab ini penulis menjabarkan metode penelitian yang diterapkan untuk menganalisa dan membandingkan kinerja algoritma Rijndael dan Twofish.

BAB IV Analisis dan Pengujian

Pada bab ini penulis akan menganalisa dan menguji kedua algoritma secara matematis, menggunakan *input* serta membandingkan performa algoritma Rijndael dan Twofish

BAB V Kesimpulan

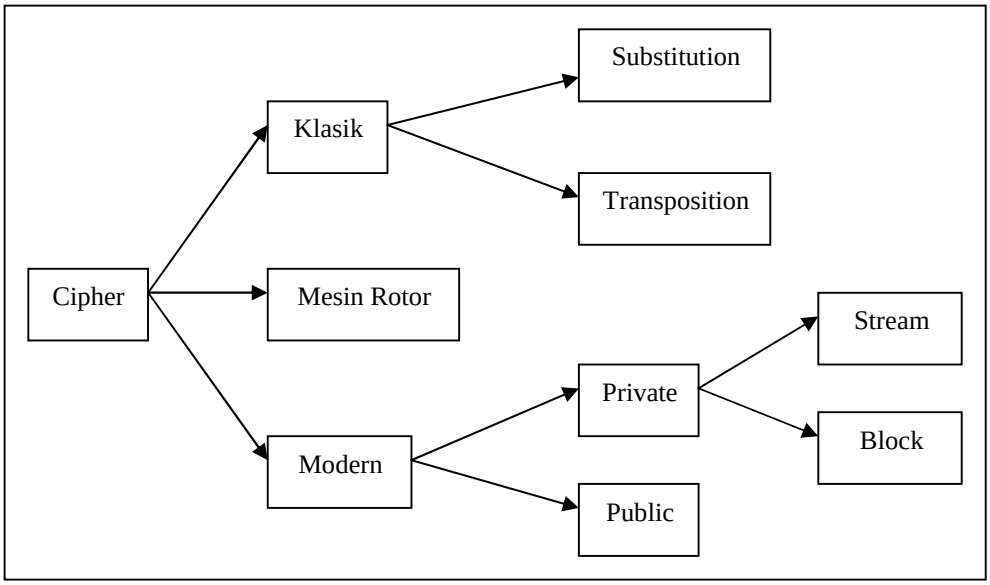
Mencakup kesimpulan dan saran analisa algoritma yang didapat dari analilsa dan pengujian kedua algoritma.

BAB II

LANDASAN TEORI

2.1 Pengertian Kriptografi

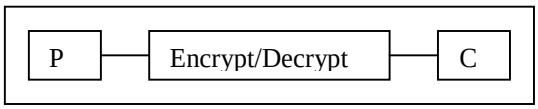
Kriptografi merupakan disiplin ilmu yang kosentrasi pada transformasi data / informasi. Ada beberapa hal yang merupakan unsur dasar kriptografi yaitu: keamanan data / informasi, keutuhan data, authentikasi dan *Non-repudiation* (diakui keabsahannya).



Gambar 2. 1 Taxonomy Algoritma Kriptography

2.1.1 Kriptografi

Kriptografi terdiri dari dua kata yaitu "*Crypto*" berarti "*secret*" (rahasia) dan "*graphy*" berarti "*writing*" (tulisan). Para pelaku atau praktisi kriptografi disebut ***cryptographer***. Sebuah algoritma kriptografi (*cryptographic algorithm*), merupakan persamaan matematik yang digunakan untuk proses enkripsi dan dekripsi, disebut ***cipher***. Biasanya kedua persamaan matematika (untuk enkripsi dan dekripsi) tersebut memiliki hubungan matematis yang cukup erat.



Gambar 2. 2 Proses Enkripi/Dekripsi

Dengan :

C= S(P)

S( )= *Encryption Function*

P=H(P)

H( ) = *Decryption Function*

Proses yang dilakukan untuk mengamankan sebuah pesan (yang disebut *plaintext*) menjadi pesan yang tersembunyi (disebut *ciphertext*) adalah enkripsi(*encryption*). *Ciphertext* adalah pesan yang sudah tidak dapat dibaca dengan mudah. Menurut ISO 7498-2, terminologi yang lebih tepat digunakan adalah "*encipher*" (Rotman, 2003).

Proses sebaliknya, untuk mengubah *ciphertext* menjadi *plaintext* , disebut dekripsi(*decryption*). Menurut ISO 7498-2, terminologi yang lebih tepat untuk proses ini adalah "*decipher*". *Cryptanalysis* adalah seni dan ilmu untuk memecahkan *ciphertext* tanpa bantuan kunci.

Perkembangan teknologi yang maju saat ini tidak dapat dilepaskan dari perkembangan ilmu pengetahuan. Kemajuan yang sangat pesat dalam industri juga didukung oleh perkembangan teknologi dan ilmu pengetahuan dari dunia pendidikan.

2.1.2 Kriptanalis

Kriptanalisis (c*ryptanalysis)* adalah seni dan ilmu untuk memecahkan *ciphertext* tanpa bantuan kunci. Kriptnalis/kriptografer (c*ryptanalyst)* adalah pelaku atau praktisi yang menjalankan kriptanalis (Newton, 1997).

2.1.3 Kriptologi

Kriptologi (*Cryptology*) merupakan disiplin ilmu yang mencakup kriptografi dan kriptanalis (Newton, 1997).

2.2 Teori Bilangan

Ada beberapa istilah yang kan digunakan, seperti *field*, *group* dan *ring*. Misal operasi biner (*) pada S*S merupakan operasi pemetaan dari S ke S.

2.2.1 Bilangan Biner dan Hexadecimal

Biner adalah bilangan yang dituliskan dalam 1 dan 0, sedangkan *hexadecimal* adalah bilang yang dituliskan dalam 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E dan F.

2.2.2 Field

Field adalah semua elemen (*biner* / *hexadecimal*) ataupun operasi (perkalian, penjumlahan, pengurangan, modulo dan lain sebagainya) yang terjadi pada operasi bilangan.

2.2.3 Group (G,o)

Group (*G*,o) adalah fungsi *G* dengan operasi “o”. Berikut ketentuan yang berlaku pada *Group*(Menez, 1996):

1. Jika a,b Є G maka a o b = c, c Є G (*closure*)
2. Jika (a o b) o c = a o (b o c) (assosiatif)
3. Misal e adalah elemen identitas maka a o e = e o a, a (identitas).
4. a o a^{-} = e (identitas)

2.2.4 Ring (R,+, x)

Ring adalah set fungsi operasi yang terdiri dari beberapa operasi, dengan “+” adalah operasi penjumlahan dan “x” adalah operasi perkalian. Beberapa ketentuan yang berlaku (Menez, 1996):

1. assosiatif : jika a x (b x c) = (a x b) x c untuk a,b dan c € R
2. Perkalian identitas : 1 x a = a x 1 = a untuk a € R
3. Terdistribusi : a x (b + c) = (a x b)+ (a x c) untuk a,b dan c € R

2.2.5 Polynomial Ring

Misal $f(x) = a_n x^n + \ldots + a_2 x^2 + ax + a_0$

Dimana $a_i$adalah € R dan n ≥ 0 , dengan a_i adalah koefisien x_i pada *f(x)* .

2.2.6 Vector

*Vecto*r adalah hasil dari perkalian baik dalam bentuk *ring*, maupun *polynomial.*

2.3 Operasi Bilangan pada kriptografi

Pada enkripsi/dekripsi terjadi operasi–opeerasi bilangan yang sangat rumit, berikut dasar–dasar operasi.

2.3.1 Penjumlahan (Bitwise XOR)

Sama halnya dengan penjumlahan biasa yang pernah kita pelajari. Tetapi pada operasi ini di kenal dengan " XOR " di notasikan dengan "⊕" dengan ketentuan $1 \oplus 1=0$, $1 \oplus 0=1$, $0 \oplus 0=0$.

Contoh :

$\{01010111\} \oplus \{10000011\}=11010100$ → Notasi *Biner*

$\{57\} \oplus \{83\}=\{d4\}$ → Notasi *Hexadecimal*

2.3.2 Pergeseran (Cyclic Shift/Bit Rotation)

Pada operasi *bit–bit* masukan digeser kekiri ataupun kekanan sesuai dengan nilai penggesernya. Misal $x=(x_1x_2x_3x_4)$ merupakan *bit–bitblock* dengan ukuran n, dan digeser kekiri dengan nilai m=2 maka nilai output $y=(x_3x_4x_1x_2)$.

2.3.3 P-BOX

P-BOX merupakan permutasi *block* n bit. Misal *block* masukan adalah x_1, x_2, … x_n dan sebagai permutasinya di spesifikasikasikan dengan i1, i2, …in maka *blockoutput* adalah x_{i1},x_{i2},…x_{in}(Rotman, 2003).

Contoh :

x = 101010 →*input*

P = 531246 → permutasi

y = 111000 →*output*

2.3.4 S-BOX

Operasi S-BOX merupakan operasi pemetaan dari *bit input* menjadi *bit output*. Pemetaan m-ke-n adalah mengambil *bit* m dari *input* dan menyesuaikan dengan nilai n pada tabel *lookup* (Canright, 2004).

Contoh :

Tabel S-BOX berikut adalah tabel *lookup* yang digunakan oleh DES, misal fungsi S-BOX dari nilai *x*=011011.

Tabel 2. 1S-BOX DES

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Pada *x*=($x_1x_2x_3x_4x_5x_6$) maka nilai x_1x_6 diidentifikasikan sebagai *index* baris sedangkan nilai $x_2x_3x_4x_5$sebagai *index* kolom. Dari nilai *x* maka *index* baris ada $(01)_2 = (1)_{10}$ dan *index* kolom $(1101)_2 = (13)_{10}$ maka nilai *y* adalah $(5)_{10} = (000111)_2$.

2.4 Rijndael

Rijndael ataupun yang lebih dikenal dengan AES (*Advance Encryption Standard*) mendukung kunci 128, 192 dan 256 *bit* merupakan kriptografi iterasi *block* yang berarti *block* di enkripsi sebelum melanjutkan *block* (*state*) selanjutnya. *Round* Rijndael terdiri dari *Addroundkey, ByteSub, ShiftRow dan MixColomn*.

2.4.1 S-Box Rijndael

Pada proses *SubByte* dibutuhkan tabel *look-up* (*S-Box*), S-Box dibangun dengan cara perkalian *inverse* pada *finite field* GF(2^8) dimana elemen {00} berarti ditujukan ke dirinya sendiri dan kemudian dijumlahkan menggunakan transformasi *affine*, yaitu:

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

Untuk 0<= i < 8, dimana b_i adalah *bit* ke-i dari *byte* b dan c_i adalah *bit* ke-i dari *byte* c dengan nilai c {63} atau {01100011}.

Dengan matrik dapat ditulis menjadi:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Dengan menggunakan kedua transformasi tersebut maka diperoleh tabel *lookup* yang disebut S-Box, berikut tabel *lookup* S-BOX dan S-BOX *inverse* pada AES.

Tabel 2. 2 S-BOX AES/Rijndael

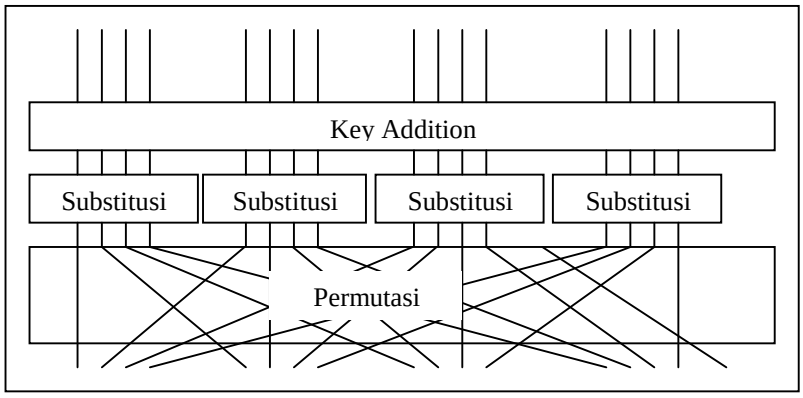
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	GF	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	52	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	DO	EF	AA	FB	43	AD	33	85	45	F9	02	7F	50	3C	9F	S
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Tabel 2. 3 S-BOX Inverse AES/Rijndael

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	9	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	4A	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	8	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92

5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	0	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	14	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	EB	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	87	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

2.4.2 Substitution-permutation Network



Gambar 2. 3 S-P Network

Substitution-permutation network merupakan salah satu model bentuk yang paling banyak digunakan *cipher*. *Rijndael* menggunakan *substitution-permutation network* sebagai struktur dasar blok. Setiap *round* terdiri tiga komponen yakni substitusi, permutasi dan *key addition* dan ketiga operasi tersebut merupakan operasi *non-linear* dan *linear*.

Substitution merupakan komponen *non-linear* yaitu operasi menyatukan *bit* pada level sub *block*, sedangkan *permutation* adalah komponen *linear* dengan cara menyebarkan atau mengacak *bit*.

2.4.2.1 AddRoundKey

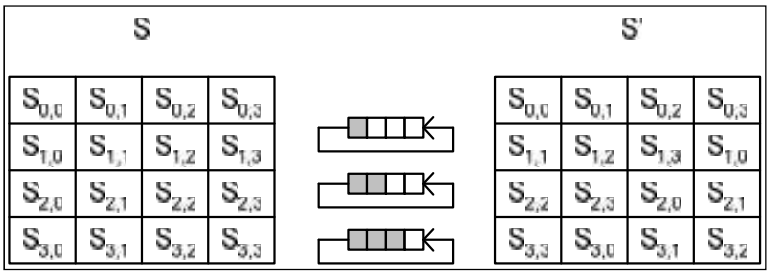
Setiap 16 *ByteState* yang di XOR-kan dengan 16 *ByteKey* , *Key* tersebut tidak akan digunakan lagi melainkan *Key* Berikutnya. Misal pada proses XOR *State* dengan *key* pada *round* awal yakni 1 – 16 maka proses XOR pada tahap kedua maka *key* yang digunakan adalah 17 – 32 begitu seterusnya.

2.4.2.2 SubByte

SubByte adalah sebuah substitusi *non-linear* yang beroperasi secara independen pada setiap *byte – bytestate*, dalam hal ini operasinya menggunakan sebuah tabel substitusi selanjutnya disebut *s-box*.

2.4.2.3 ShiftRow

ShiftRow pada dasarnya adalah proses pergeseran *bit* paling kiri dipindahkan menjadi *bit* paling kanan (rotasi *bit*), transformasi ini merupakan bentuk permutation pada level row (baris). Transformasi ini diterapkan pada baris 2, baris 3, dan baris 4. Baris 2 akan mengalami pergeseran *bit* sebanyak satu kali, sedangkan baris 3 dan baris 4 masing-masing mengalami pergeseran *bit* sebanyak dua kali dan tiga kali. Contoh:



Gambar 2. 4 Transformasi ShifRow

2.4.2.4 MixColomn

Mix coloumn merupakan bentuk permutasi pada coloumn (baris), pada proses ini terdapat dua proses rumit yakni proses perkalian *matrix*. Setiap elemen pada satu kolom *state* dikalikan dengan suatu *polynomial* tetap dan setiap hasil perkalian di-XOR-kan, secara seacara matematis ditulis sebagai berikut : $a(x) = \{03\}x^3 \oplus \{01\}x^2 \oplus \{01\}x \oplus \{02\}$.

Matrik ini adalah:

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

Gambar 2. 5 Matrix Pengali a(x)

Perkalian 4 nilai kolom *matrix* pengali pertama dengan 4 nilai baris *state* pertama, kemudian hasilnya di-XOR-kan untuk menghasilkan 1 *byte*. Dan 4 nilai kolom matrix pengali kedua dengan 4 nilai baris *state* kedua, kemudian hasilnya di-XOR-kan untuk menghasilkan 1 *byte*, begitu seterusnya sampai pada perkalian terakhir. Secara matematis dapat ditulis menjadi:

$$b1 = (b1 * 2) xor (b2 * 3) xor (b3 * 1) xor (b4 * 1)$$
$$b2 = (b1 * 1) xor (b2 * 2) xor (b3 * 3) xor (b4 * 1)$$
$$b3 = (b1 * 1) xor (b2 * 1) xor (b3 * 2) xor (b4 * 3)$$
$$b4 = (b1 * 3) xor (b2 * 1) xor (b3 * 1) xor (b4 * 2)$$
dst
$$b16 = (b13 * 3) xor (b14 * 1) xor (b15 * 1) xor (b16 * 2)$$

2.4.3 Pengembangan Kunci

Sebelum proses enkripsi dan dekripsi maka *key* harus dikembangkan. Dan *key* ini yang akan digunakan pada proses *AddRoundKey*. Setiap proses *AddRoundKey* selalu memanggil bagian *key* yang berbeda kemudian *key* ini akan di-XOR-kan dengan *state*, untuk ini dibutuhkan suatu *key* yang besar dan juga dibutuhkan waktu yang lebih untuk memanggil kunci yang berbeda. Apabila kita akan proses enkripsi pada algoritma Rijndael dengan *key* 128 maka dibutuhkan 16*(jumlah *round* +1) yakni 176 *key* .

Setiap tahapan menambahkan 4 *byte* pada *key* expansion. Dengan pengeculian untuk tahapan awal dengan mengambil 4 *byte* sebelumnya sebagai *Input*(masukan) dan akan menghasilkan 4 *byte*. Pada operasi ini tidak semua fungsi akan dipangggil untuk setiap tahapannya. Fungsi – fungsi yang mendukung pengembangan kuncisecara berurutan adalah (*rotword*,*subword*, *rcon*, *ek* dan *k*) dan 4 fungsi selalu dipanggil kecuali fungsi *k* untuk setiap 4 *round* untuk *key* = 16 *byte*, 6 *round* untuk *key* = 24 *byte*, dan 8 *round* untuk *key* = 32 *byte*.

Fungsi akan dipanggil jika *key* 32 *byte* dan fungsi memanggil fungsi *subword* disetiap 8 tahapan dimulai pada tahapan ke-13.

Berikut proses – proses *key expansion*:

1. *RotWord* (4 *byte*)

Fungsi ini merupakan fungsi sama dengan *ShiftRow* yakni fungsi pergeseran (*Cyclic Shift*)

1, 2, 3, 4 → 2, 3, 4, 1

2. *SubWord* (4 *byte*)

Pada fungsi ini menggunakan tabel *S-BOX* untuk setiap fungsi *SubWord* (Perhatikan proses transformasi *SubByte* yang telah dijabarkan pada bab sebelumnya).

3. *Rcon* ((*Round* /(*Key* Size/4))-1)

Berikut nilai – nilai Rcon yang digunakan untuk pengembangan kunci AES.

Rcon (0) = 01000000

Rcon(1) = 02000000

Rcon(2) = 04000000

Rcon(3) = 08000000

Rcon(4) = 10000000

Rcon(5) = 20000000

Rcon(6) = 40000000

Rcon(7) = 80000000

Rcon(8) = 1B000000

Rcon(9) = 36000000

Rcon(10) = 6C000000

Rcon(11) = DB000000

Rcon(12) = AB000000

Rcon(13) = 4D000000

Rcon(14) =9A000000

Contoh:

Untuk *Key* = 16 *byte*, fungsi *Rcon* pertama kali akan dipanggil pada tahapan ke-4 Rcon(4/(16/4))-1)=0, dari tabel didapat *Rcon*(0)=01000000,

Untuk *Key* = 24 *byte*, fungsi *Rcon* pertama kali akan dipanggil pada tahapan ke-6 Rcon(6/(24/4))-1)=0, Rcon(0)=01000000, dan

Untuk *Key* = 32 *byte*, Rcon akan dipanggil pada tahapan ke-8 dimana Rcon(0)=01000000.

4. Offset EK dan *K*

Fungsi offset *EK* hanya sebagai pengganti, misal offset=0 maka fungsi ini akan memberikan nilai *key* expansion = 0,1,2,3 sedangkan *offset* K juga sebagai pengganti nilai *key* expansion menjadi 0,1,2,3 apabila nilai *offset* = 0.

2.5 Twofish

Twofish merupakan algoritma kriptografi *block* dengan ukuran 128 *bit* dan mendukung kunci 128, 192 dan 256 *bit*, twofish menggunakan 16-*round feistel network* dimana didalamnya terdapat fungsi F yang berfungsi untuk membangun 4 buah kunci dalam 8-*bits-box* secara *bijective*, MDS (*maximum distance separable matrix*), PHT (*pseudo-Hadamard transformation*), rotasi *bit* dan penjadwalan kunci (*key schedule*). Twofish juga menggunakan 2 proses *whitening*, ini digunakan untuk meng-XOR-kan *plaintext* dengan kunci (K0, K1, K2 dan K3) proses ini disebut *input-whitening* dan proses meng-XOR-kan *ciphertext* dengan kunci (K4, K5, K6 dan K7) kemudian disebut *output–whitening*.

Plaintext 128-*bit* dipotong menjadi 4 *word* (Q0, Q1, Q2 dan Q3) masing - masing berukuran 32-*bit word*. Pada tahapan *input* whitening 4 *plaintext* yang dipotong di-XOR dengan kunci (K0, K1, K2 dan K3), selanjutnya hasil dari *input-whitening* diproses menggunakan 16-*round feistel network*. Pada 16-*round feistel network*, Q1 di rotasikan 8-*bit* kekiri kemudian Q0 dan Q1 digunakan sebagai *input* pada fungsi *g*. Pada fungsi *g* terdapat 2 fungsi utama yang terjadi yakni *s-box* dan MDS, 2 output dari fungsi *g* di kombinasi menggunakan PHT selanjutnya masing - masing di tambahkan dengan kunci kemudian masing - masing di-XOR dengan Q2 dan Q3 (salah satunya di rotasi kekiri 1 *bit* dan satunya di rotasi

kekanan 1 *bit*) kemudaian output diswap (Q0 ↔ Q2 dan Q1 ↔ Q3). Dan pada round akhir *output* di *swap* kembali selanjutnya dilakukan *output-whitening*.

2.5.1 Feistel Network

Feistel network adalah metode umum untuk mentransformasi suatu fungsi menjadi bentuk permutasi. Bagian paling fundamental dari *fiestel network* adalah fungsi F: sebuah pemetaan *key-dependent* dari suatu *inputstring* menjadi *output string*. Dalam Twofish dilakukan *fiestel network* sebanyak 16 kali.

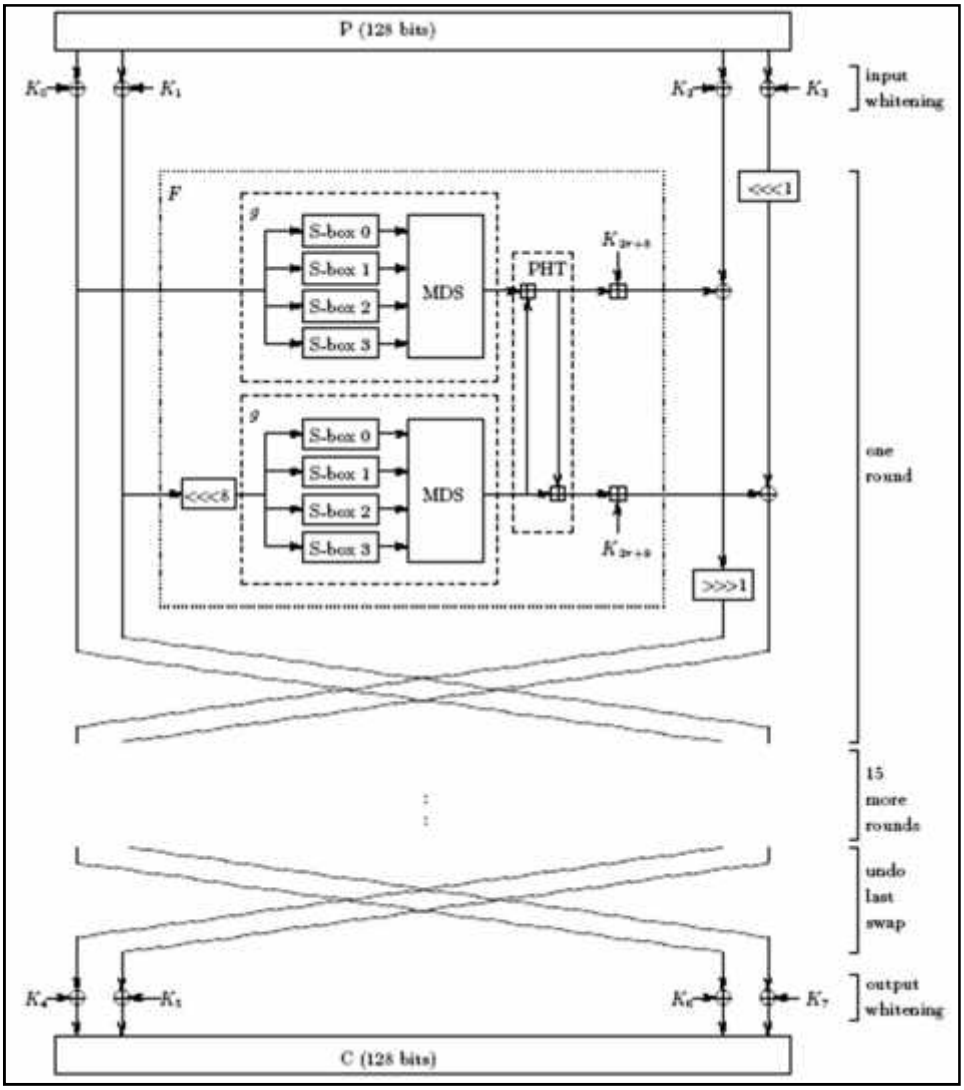
Fiestel network pada twofish terdiri dari *input Whitening*, *S-box*, transformasi *pseudo hadamard*, *output* dan *output whitening*.

2.5.1.1 Fungsi *F*

Pondasi dasar dari *feistel network* adalah fungsi *F*, yaitu suatu permutasi *key-dependent* terhadap nilai 64-*bit*. Fungsi *F* memerlukan tiga buah argumen, dua *Inputword* R_0 dan R_1, dan bilangan bulat *r* yang digunakan untuk memilih *subkey* yang bersesuaian. R_0 dilewatkan fungsi *g*, yang menghasilkan T_0. R_1 dirotasikan dalam sebuah PHT dan dua *word* dari *key* yang dikembangkan kemudian ditambahkan kepadanya.

$$T_0 = g(R_0)$$
$$T_1 = g(ROL(R_1,8))$$
$$F_0 = (T_0 + T_1 + K_{2r+8}) \bmod 2^{32}$$
$$F_1 = (T_0 + T_1 + K^{2r+9}) \bmod 2^{32}$$

Dimana (F_0, F_1) merupakan hasil dari *F*, *ROL* adalah rotasi ke kiri terhadap R_1sejauh 8 *bit*. Fungsi *F* selalu *non-linear* dan kemungkinan non *surjektif*, yaitu bahwa tidak semua *output* yang dimungkinkan berada dalam ruang *output* dapat terjadi semua.



Gambar 2. 6 Feistel Network Twofish

2.5.1.2 Fungsi *g*

Fungsi g adalah jantung algoritma Twofish yang merupakan komponen utama dari fungsi F. Fungsi g menggunakan 32 *bit Vector* X dan 64 *bit vector* L untuk memproduksi 32 *bit output* Z=g(X,L). dimana 32 *bit input* dibagi menjadi 4 *byte* dimana setiap *byte* dipetakan menggunakan *key-dependent* S-Box dimana *output* di iterprestasikan sebagai 4 *vector* yang selanjutnya di *multiply*kan dengan *matrix* MDS dengan ukuran 4 x 4. Adapun fungsi *g* di jabarkan sebagai berikut:

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = [MDS] \bullet \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

Dengan,

$x_i = \left[x / 2^{8i} \right] \bmod 2^8$
$y_i = s_i \left[x_i \right]$
$i = 0,1,2,3$

Dengan matrix MDS:

$$MDS = \begin{pmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{pmatrix}$$

2.5.1.3 S-Box

S-Box selalu digunakan pada proses enkripsi/dekripsi maupun penjadwalan kunci, *S-box* adalah operasi substitusi *table-driven non-linear* yang digunakan dalam blok*cipher*. *S-box* bervariasi antara setiap ukuran *input*dan ukuran *output*nya, dan bisa diciptakan secara random atau dengan algoritma. Pada *twofish S-box* diproduksi menggunakan 4 buah permutasi 8-by-8-*bit* dan material *key* , yakni:

$s_0(x) = q_1 \left[q_0 \left[q_0 [x] \oplus s_{0,0} \right] \oplus s_{1,0} \right]$
$s_1(x) = q_1 \left[q_0 \left[q_1 [x] \oplus s_{0,1} \right] \oplus s_{1,1} \right]$
$s_2(x) = q_0 \left[q_1 \left[q_0 [x] \oplus s_{0,2} \right] \oplus s_{1,2} \right]$
$s_3(x) = q_0 \left[q_1 \left[q_1 [x] \oplus s_{0,3} \right] \oplus s_{1,3} \right]$

2.5.1.4 MDS (Maximum Distance Separable)

MDS adalah sebuah pemetaan linear dari elemen *field a* ke elemen *field b* yang menghasilkan campuran dari *vector a + b*. Dengan kata lain “Distance”

adalah jumlah element yang berbeda antara dua vector yang berbeda yang dihasilkan oleh MDS paling kurang b+1. Pemetaan MDS bisa direpresentasikan oleh matriks MDS yang terdiri dari *a* x *b*. *Twofish* menggunakan matriks MDS 4x4.

2.5.1.5 PHT (Pseudo-Hadamard transformation)

PHT adalah operasi sederhana yang bekerja dengan cepat pada *input*. dan PHT (32 *bit*) didefinisikan sebagai :

$A_0 = a + b \bmod 2^{32}$

$B_0 = a + 2b \bmod 2^{32}$

Twofish menggunakan PHT (32 *bit*)untuk melakukan *mixing* terhadap *output* nya dengan memparalelkan dua buah fungsi g (32 *bit*).

2.5.1.6 Bit Rotation

Pada *feistel network,* twofish terdapat fungsi rotasi (ROR rotasi kekanan dan ROL rotasi kekiri). Fungsi dilakukan pada setiap *round* yakni rotasi 8 *bit* kekiri pada Q1 sebagi *input* fungsi *g*, rotasi 1 *bit* kekiri Q3 sebelum peng-XOR-an dengan salah satu *output* PHT dan rotasi 1 *bit* kekanan pada hasil peng-XOR-an *output* PHT dengan Q2.

2.5.2 Whitening

Whitening merupakan teknik meng-XOR-kan *key*material dengan *plaintext*, pada twofish terdapat proses dua *whitening* yakni *input-whitening* dan *output-whitening*. Proses whitening terjadi diluar 16-*round feistel network*.

2.5.2.1 *Input* Whitening

Input-whitening dilakukan hanya pada tahapan awal yaitu dengan meng-*XOR*-kan *plaintext* dengan kunci.

2.5.2.2 Output Whitenning

Apabila sebelumnya terdapat *input-whitening* maka pada tahapan akhir *input* (Q0, Q1, Q2 dan Q3) juga di-XOR dengan kunci.

2.5.3 Pengembangan Kunci

Key schedule adalah suatu cara dimana *bit-bitkey* diubah menjadi *key -key* bulat yang dapat digunakan oleh *chiper*. Twofish memerlukan material *key* yang sangat banyak, dan memiliki *key schedule* yang rumit. Untuk memudahkan analisis, *key schedule* menggunakan fungsi pembulatan biasa.

Twofish memakai 40 *wordkey* yakni $k_0,k_1,\ldots,k_{39}$ dengan panjang *key* N=128, N=192 dan N=256.*Key* M terdiri dari 8k *byte* yakni $m_0, m_1,\ldots, m8_{k-1}$ dengan k=N/64. *Byte* di *convert* (ubah) kedalam 2k yang masing – masing berukuran 32 *bit*.

$$M_i = \sum_{j=0}^{3} m_{4i+j}.2^{8j} \text{ dengan } i = 0,\ldots,2k\text{-}1.$$

Dan pada kedua vektor berikut,

$$M_e = \left(M_0, M_2, ..., M_{2k-2}\right)$$
$$M_o = \left(M_1, M_3, ..., M_{2k-1}\right)$$

Vektor ketiga diperoleh dari *expansion key* yakni perkalian vektor sebelumnya dengan *matrix* RS.

$$\begin{pmatrix} s_{i,0} \\ s_{i,1} \\ s_{i,2} \\ s_{i,3} \end{pmatrix} = \left(\begin{array}{c|c|c|c:c|c|c|c} 01 & A4 & 55 & 87 & 5A & 58 & DB & 9E \\ A4 & 56 & 82 & F3 & 1E & C6 & 68 & E5 \\ 02 & A1 & FC & C1 & 47 & AE & 3D & 19 \\ A4 & 55 & 87 & 54 & 58 & DB & 9E & 03 \end{array}\right) \bullet \begin{pmatrix} m_{8i} \\ m_{8i+1} \\ m_{8i+2} \\ m_{8i+3} \\ m_{8i+4} \\ m_{8i+5} \\ m_{8i+6} \\ m_{8i+7} \end{pmatrix}$$

Atau,

$$S_i = \sum_{j=0}^{3} S_{i,j}.2^{8j} \quad \text{dengan I = 0,…,2k-1}$$

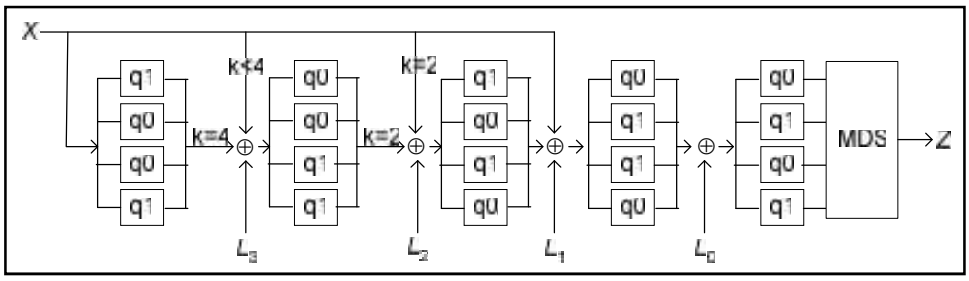
$$RS = \left(\begin{array}{c|c|c|c:c|c|c|c} 01 & A4 & 55 & 87 & 5A & 58 & DB & 9E \\ A4 & 56 & 82 & F3 & 1E & C6 & 68 & E5 \\ 02 & A1 & FC & C1 & 47 & AE & 3D & 19 \\ A4 & 55 & 87 & 54 & 58 & DB & 9E & 03 \end{array}\right)$$

2.5.3.1 Fungsi *h*

Fungsi ini digunakan untuk memproduksi I *word* dengan ukuran 32 *bitword* dari X dan L (L_0,…,L_{k-1}), fungsi terjadi sebanyak k (k=N/64). Pada setiap stagenya 4 *byte* di masukan pada *S-box* kemudian di XOR kan dengan L dan terakhir *byte* dimasukan ke *S-box* dikalikan dengan matrix MDS. Jadi *twofish* selalu pada stage ini *word* di bagi menjadi *byte – byte*.

$$l_{i,j} = \left[L_i / 2^{8j}\right] \bmod 2^8$$
$$x_j = \left[X / 2^{8j}\right] \bmod 2^8$$

Dengan i=0, …, k-1 dan j=0, …,3



Gambar 2. 7 Struktur S-Box Twofish

Dari gambar dijelaskan bahwa:

Jika k = 4

$$y_{3,0} = q_1\lfloor y_{4,0} \rfloor \oplus l_{3,0}$$
$$y_{3,1} = q_0\lfloor y_{4,1} \rfloor \oplus l_{3,1}$$
$$y_{3,2} = q_0\lfloor y_{4,2} \rfloor \oplus l_{3,2}$$
$$y_{3,3} = q_1\lfloor y_{4,3} \rfloor \oplus l_{3,3}$$

Jika k ≥ 3

$$y_{2,0} = q_1\lfloor y_{3,0} \rfloor \oplus l_{3,0}$$
$$y_{2,1} = q_1\lfloor y_{3,1} \rfloor \oplus l_{3,1}$$
$$y_{2,2} = q_0\lfloor y_{3,2} \rfloor \oplus l_{3,2}$$
$$y_{2,3} = q_0\lfloor y_{3,3} \rfloor \oplus l_{3,3}$$

Dari persamaan tersebut diperoleh:

$$y_0 = q_1\lfloor q_0\lfloor q_0\lfloor y_{2,0} \rfloor \oplus l_{1,0} \rfloor \oplus l_{0,0} \rfloor$$
$$y_1 = q_0\lfloor q_0\lfloor q_1\lfloor y_{2,1} \rfloor \oplus l_{1,1} \rfloor \oplus l_{0,1} \rfloor$$
$$y_2 = q_1\lfloor q_1\lfloor q_0\lfloor y_{2,2} \rfloor \oplus l_{1,2} \rfloor \oplus l_{0,2} \rfloor$$
$$y_0 = q_0\lfloor q_1\lfloor q_1\lfloor y_{2,3} \rfloor \oplus l_{1,3} \rfloor \oplus l_{0,3} \rfloor$$

Hasil dari fungsi (*Z*) ini adalah vektor (y_i) dikalikan dengan *matrix* MDS dengan q0 dan q1 adalah permutasi pada 8 *bit*.

2.5.3.2 Key-dependent S-Box

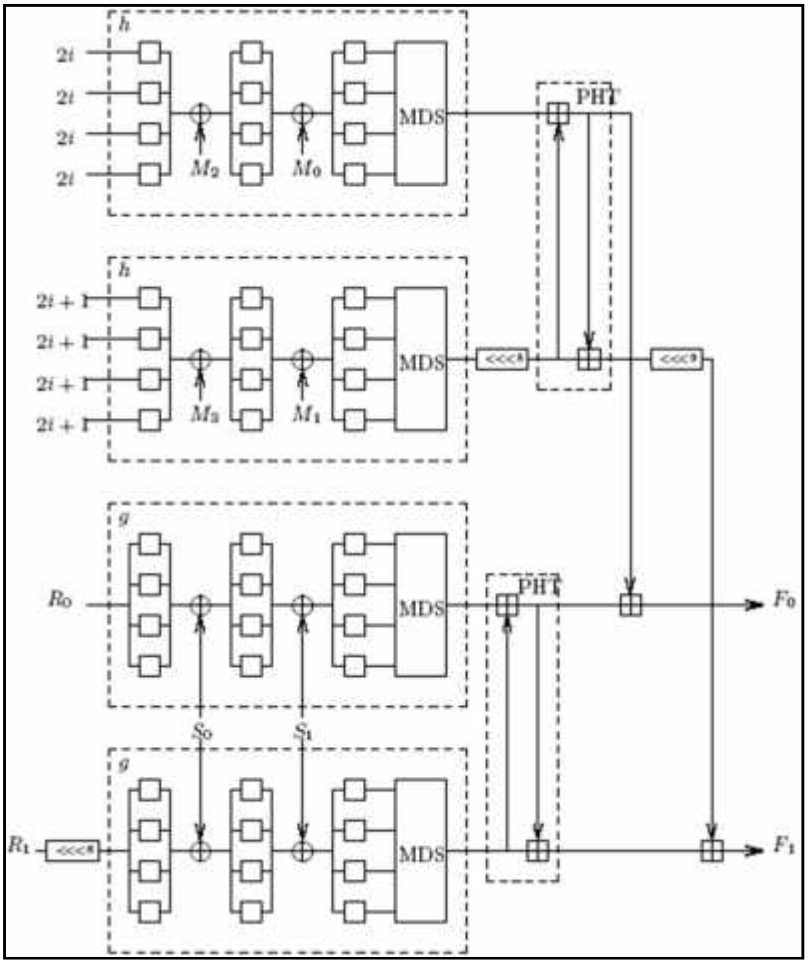
Pada pengembangan kunci, twofish juga mengunakan *S-box*. Berikut *S-box* yang digunakan untuk $q_{0:}$

$$t_0 = [8\ \ 1\ \ 7\ \ D\ \ 6\ \ F\ \ 3\ \ 2\ \ 0\ \ B\ \ 5\ \ 9\ \ E\ \ C\ \ A\ \ 4]$$
$$t_1 = [E\ \ C\ \ B\ \ 8\ \ 1\ \ 2\ \ 3\ \ 5\ \ F\ \ 4\ \ A\ \ 6\ \ 7\ \ 0\ \ 9\ \ D]$$
$$t_2 = [B\ \ A\ \ 5\ \ E\ \ 6\ \ D\ \ 9\ \ 0\ \ C\ \ 8\ \ F\ \ 3\ \ 2\ \ 4\ \ 7\ \ 1]$$
$$t_3 = [D\ \ 7\ \ F\ \ 4\ \ 1\ \ 2\ \ 6\ \ E\ \ 9\ \ B\ \ 3\ \ 0\ \ 8\ \ 5\ \ C\ \ A]$$

Dan, untuk q_1 tabel *S-box* adalah :

$$t_0 = [2 \quad 8 \quad B \quad D \quad F \quad 7 \quad 6 \quad E \quad 3 \quad 1 \quad 9 \quad 4 \quad 0 \quad A \quad C \quad 5]$$
$$t_1 = [1 \quad E \quad 2 \quad B \quad 4 \quad C \quad 3 \quad 7 \quad 6 \quad D \quad A \quad 5 \quad F \quad 9 \quad 0 \quad 8]$$
$$t_2 = [4 \quad C \quad 7 \quad 5 \quad 1 \quad 6 \quad 9 \quad A \quad 0 \quad E \quad D \quad 8 \quad 2 \quad B \quad 3 \quad F]$$
$$t_3 = [C \quad 9 \quad 5 \quad 1 \quad C \quad 3 \quad D \quad E \quad 6 \quad 4 \quad 7 \quad F \quad 2 \quad 0 \quad 8 \quad A]$$

2.5.3.3 Pengembangan kunci K_j



Gambar 2. 8 Struktur Fungsi-h pada pengembangan kunci Twofish

Twofish menggunakan fungsi *h* untuk mengembangan atau memperluaskan kunci.

$$\circ = 2^{24} + 2^{16} + 2^8 + 2^0$$
$$A_i = h(2i\circ, M_e)$$
$$B_i = ROL(h((2i+1)\circ, M_o), 8)$$
$$K_{2i} = (A_i + B_i) \bmod 2^{32}$$
$$K_{2i+1} = ROL((A_i + 2B_i) \bmod 2^{32}, 9)$$

Konstanta $\circ$ digunakan untuk menduplikasikan *byte*. *i* $\circ$ terdiri dari 4 *byte* yang sama Untuk setiap *i*-nya.Fungsi h (*word*) Nilai *byte* $A_i$adalah 2i dan M_e. B_i dihasilkan menggunakan rotasi nilai B*i*. Kemudian nilai A_i dan B_i di kombinasikan menggunakan PHT. Kemudian hasil PHT tersebut dirotasikan dengan 9 *bit*.

2.5.3.4 Permutasi q0 dan q1

Permutasi ini merupakan permutasi dalam 8 *bit*. Untuk setiap permutasi, q0 dan q1 dibangun dari 4 *byte* yang berbeda dengan *input*nya adalah *x* dan *Output* adalah *y*.

a_0, b_0 = [x/16], x mod 16
a_1 = $a_0 \oplus$ b0
b_1 = $a_0 \oplus \text{ROR}_4(b_0, 1) \oplus 8a_0 \bmod 16$
a_2, b_2 = $t_0[a_1], t_1[b_1]$
a_3 = $a_2 \oplus b_2$
b_3 = $a_2 \oplus \text{ROR}_4(b_2, 1) \oplus 8a_2 \bmod 16$
a_4, b_4 = $t_2[a_3], t_3[b_1]$
y = $16b_4 + a_4$

Dengan ROR_4 adalah fungsi rotasi dalam 4 *bit*, pertama *byte* di bagi dua (a dan b), data yang telah di bagi di *mixing* menggunakan tabel *S-box.*

2.6 Set Instruksi

Pada level instruksi disini akan dibahas mengenai set-set instruksi yang digunakan enkripsi symetris khususnya Rijndael dan Twofish pada *finite machine*. Instruksi–instruksi sangat mempengaruhi performance, *time recovery* (jumlah putaran ketika sebuah unit fungsional digunakan oleh instruksi hingga unit

fungsional tersebut dapat digunakan oleh intruksi lainnya dan *latency* (jumlah *cycle* ketika hasil eksekusi intruksi dapat digunakan oleh oleh instruksi lainnya).

Pada analisa ini saya mengasumsikan bahwa set instruksi, *latency* instruksi, set register, arsitektur *cache* sebagai *real machine* di kelompokkan menjadi 2 unit fungsional yakni MEM dan ALU.

MEM merupakan set instruksi untuk *load* dan *store* dan ALU adalah set instruksi *AND, XOR, ADD , SUB, OR, NOT, EXTRACT BYTE, SHIFT, dan ROTATE*.

Tabel 2. 4 Set Instruksi, Recovery Time dan Latency

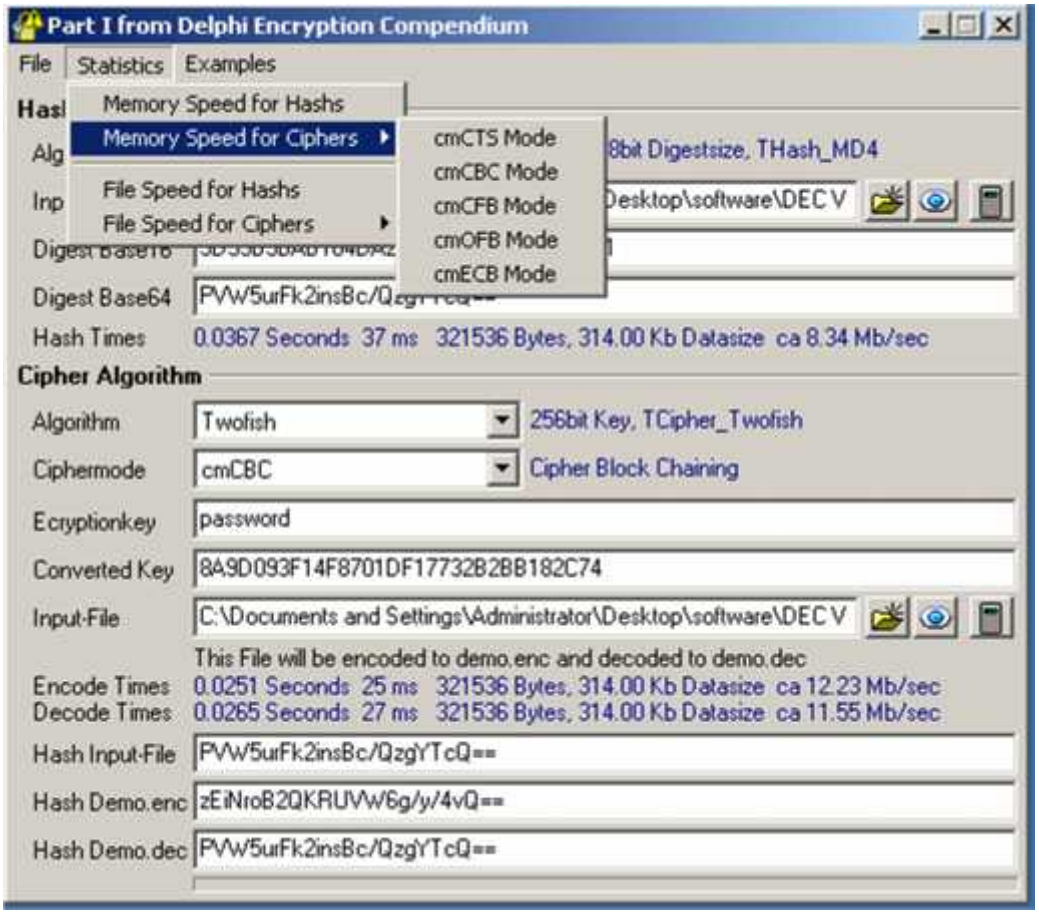
Jenis Instruksi	Unit Fungsional	Recovery Time (Cycle)	Latency (Cycle)
LOAD	*MEM*	1	3
STORE	*MEM*	1	1
ADD , SUB	*ALU*	1	1
AND, OR, XOR. NOT	*ALU*	1	1
BYTE EXTRACT (BXT)	*ALU*	1	1
SHIFT, ROTATE	*ALU*	1	1

2.7 Delphi Encryption Compedium

Delphi Encryption Compedium adalah program enkrispsi file berbasis *GUI (Graph User Interface)*yang dikembangkan oleh Hagen

Reddmannhttp://www.torry.net/, adapun spesifikasi program tersebut adalah sebagai berikut :

1. Mencakup algoritma fungsi Checksum yakni : *CRC32, XOR32-bit, XOR16-bit, CRC16-CCITT, CRC16-Standard*
2. 23 algoritma *hash* : *MD4, MD5, SHA (other Name SHS), SHA1, RipeMD128, RipeMD160, RipeMD256, RipeMD320, Haval (128, 160, 192, 224, 256) with Rounds, Snefru, Square, TigerSapphire II (128, 160, 192, 224, 256, 288, 320)*
3. 40 algoritma kriptografi: *Gost, Cast128, Cast256, Blowfish, IDEA, Mars, Misty 1, RC2, RC4, RC5, RC6, FROG, Rijndael, SAFER, SAFER-K40, SAFER-SK40,SAFER-K64, SAFER-SK64, SAFER-K128, SAFER-SK128, TEA, TEAN, Skipjack, SCOP, Q128, 3Way, Twofish, Shark, Square, Single DES, Double DES, Triple DES, Double DES16, Triple DES16, TripleDES24, DESX, NewDES, Diamond II, Diamond II Lite, Sapphire II.*
4. 2 algoritma generator *Standard Random Generator, Linear Feedback Shift Register* RNG dengan variable waktu dari 2^64-1 hingga 2^2032-1.
5. 6 format teks: *Hexadecimal, MIME Base 64, Plain, RFC1760 Six Word, UU Coding, XX Coding*
6. Standar model algoritma ECB, CBC, CFB, OFB dan CTS
7. Semua algoritma enkripsi dan hash dapat di uji secara bersama yang dapat digunakan untuk menganalisa penggunaan memori dan performa terhadap file uji secara statistik pada mode enkripsi.

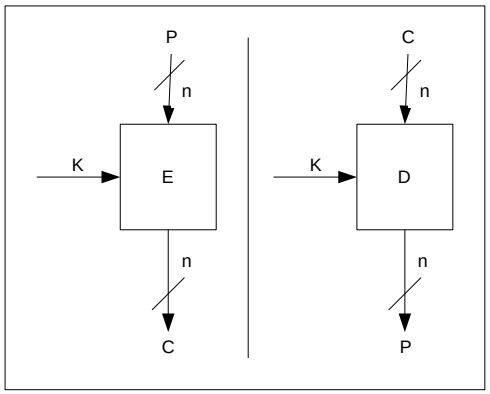


Gambar 2. 9 Tampilan Delphi Encryption Compendium

2.8 Model Kriptografi

2.8.1 Electronik Code Book (ECB)

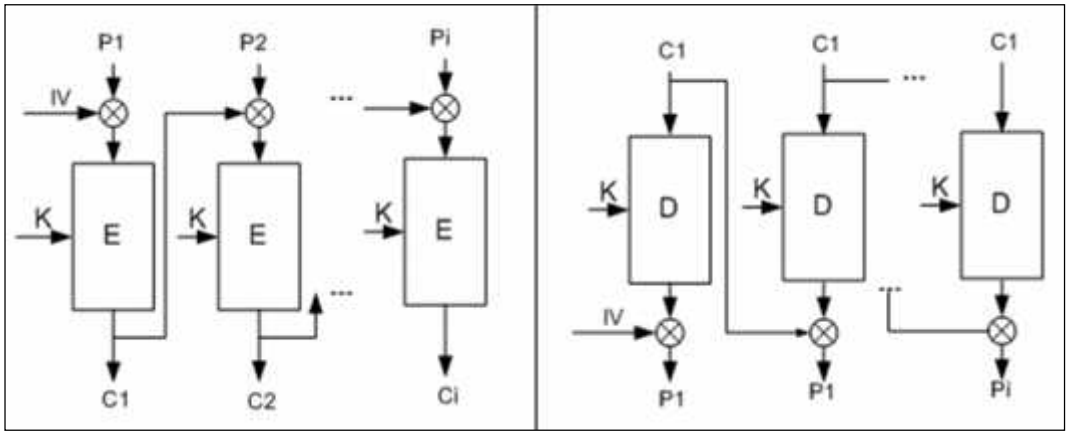
ECB Merupakan model kriptografi yang paling simple. Dimana setiap pesan (*plaintext*) di bagi menjadi *block–block* yang independen. Kekurangan pada model ini adalah pengidentifikasi (*key*) *block plaintext* di enkript untuk mengidentifikasi *ciphertext* atau dengan kata lain untuk proses pengkodean *block plaintext* dan *ciphertext* menggunakan *key* yang sama. Apabila terdapat kesalahan pada *plaintext* Ci akan menghasilkan *plaintext* yang salah, karena itu model ini sangat tidak cocok untuk data yang besar tetapi sangat ideal untuk data yang kecil.



Gambar 2. 10 Electronik Code Book

2.8.2 Chiper Block Chaining (CBC)

Pada model ini setiap *block plaintext* di XOR kan dengan *block cipher* sebelumnya. Pada model ini *ciphertext* dipengaruhi oleh *plaintext* yang di XOR kan.



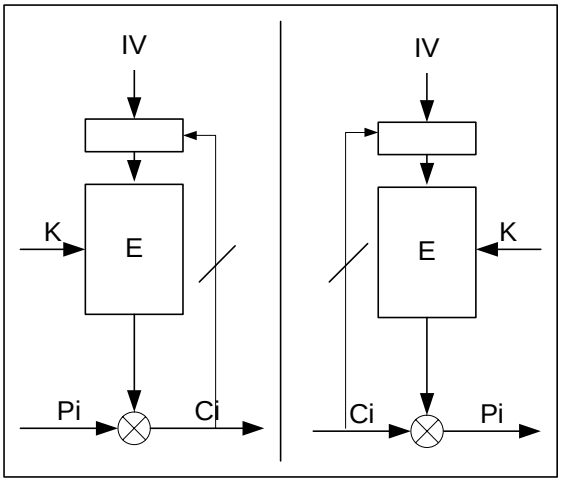
Gambar 2. 11 Chiper Block Chaining

2.8.3 Chiper Text Stealing (CTS)

Model ini adalah variant dari CBC perbedaannya adalah *block plaintext* di-*multiply*-kan dengan *stream plaintext* sebelumnya.

2.8.4 Chiper FeedBack CFB

Pada model ini biasa digunakan pada stream, jika *plaintext* merupakan stream dari karakter - karakter j-*bit* (pada umumnya jumlahnya 8 *bit* untuk karakter ASCII) dan ini akan dipersiapkan untuk menghasilkan output potongan–potongan j-*bit* pula.



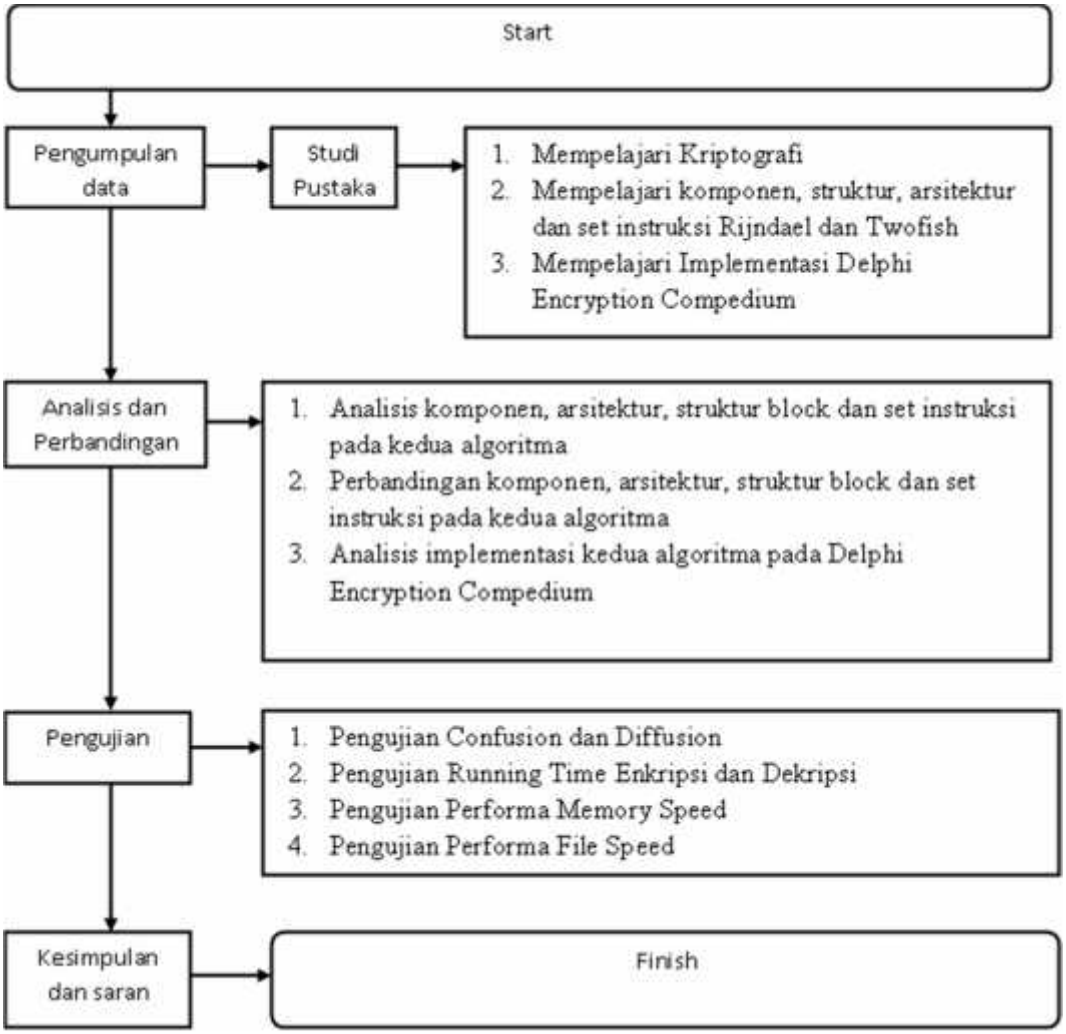
Gambar 2. 12 Chiper FeedBack

2.8.5 Output FeedBack (OFB)

Model ini hampir sama dengan CFB, tetapi menggunakan j-*bit* dari proses enkripsi dan akan memasukan kembali siklus enkripsi, biasanya digunakan sebagai sebuah *pseudorandom number generator* sehingga apabila ada kerusakan pada proses ini tidak akan mempengaruhi *ciphertext* .

BAB III

METODE PENELITIAN



Gambar 3.1 Flowchart MetodePenelitian

Metodologi penelitian adalah cara yang digunakan dalam memperoleh berbagai data untuk diproses menjadi informasi yang lebih akurat sesuai permasalahan yang akan diteliti. Metodologi yang digunakan dalam penelitian tugas akhir yang berjudul ”Analisa Algoritma Rijndael dan Twofish” secara garis besar meliputi pengumpulan data yakni kegiatan mempelajari algoritma kriptografi dari berbagai sumber literatur atau disebut juga dengan studi pustaka,

analisis dan perbandingan algoritma meliputi kegiatan menganalisa dan membandingkan teknologi kedua algoritma seperti komponen *block*, arsitektur *block*, struktur *block* dan set instruksi pada tahapan ini juga dilakukan analisa terhadap implementasi kedua algoritma pada Delphi Encryption Compedium, pada tahapan pengujian dilakukan serangkaian performa yang meliputi proses enkripsi dan dekripsi, memory speed dan file speed, pada tahapan akhir membuat kesimpulan dari hasil penelitian yakni hasil analisa dan perbandingan dengan tujuan mendiskripsikan kelebihan dan kekurangan kedua algoritma baik itu dari arsitektur *block*, struktur *block*, komponen *block* dan set instruksi serta implementasinya pada Delphi Encryption Compedium, berikut langkah-langkah penelitian yang dilakukan;

3.1 Pengumpulan Data

Metode penelitian dilakukan dengan mempelajari algoritma Rijndael dan twofish yang meliputi struktur *block*, komponen *block*, arsitektur *block* dan set instruksi daristudipustakayang di gunakan pada kedua algoritma pada proses enkripsi ataupun dekripsi.

3.2 Analisis dan perbandingan Rijndael dan Twofishpada DEC

Metode penelitian dilakukan dengan mempelajari dan menganalisa implementasi algoritma Rijndael dan Twofish pada Delphi Encryption Compedium. Untuk mempelajari implementasi kedua algoritma penulis menggunakan bahasa pemrograman Borland Delphi 7.0.

Penulis mempelajari implementasi struktur Rijndael dan Twofish pada Delphi Encryption Compedium.

3.3 Spesifikasi Pengujian

UjiimplementasialgoritmaRijndaeldanTwofishpada Delphi Encryption Compediumdilakukanmengunakanobjek yang identikuntuksetiappengujian.

1. Processor : AMD Athlon (tm) II X2 245 *processor* ~2.9 Ghz (3.0 Ghz)
2. Memori : 894 MB (1 Ghz Shared)
3. Sistem Operasi : Windows XP Professional Versi 2002 *Service Pack* 3
4. Bahasa Pemrograman : Borland Delphi 7.0
5. *Software* Penguji: Delphi Encryption Compedium versi 5.2
6. File Uji :

Tabel 3.1 File Input Pengujian

Data	Ukuran (KB)	Keterangan
Lusca_Upgrade_Ubuntu	5	*File* dokumen dengan format TXT
Config	100	*File* gambar dengan format JPG
Excel2002VBA	610	*File* dokmen dengan format PDF
LUSCA	1041	*File* dokumen dengan format DOCX
Ebiet - Titip rindu buat ayah	5112	*FileAudio* dengan format MP3
Syura - Nazam Berkasih	15849	*FileVideo* dengan Format MP4

3.4 Pengujian Rijndael dan Twofish pada DEC

Pada metodeini penulis melakukan pengujian *running time* dan *running resource* proses untuk mengetahui penggunaan *space* (waktudan*resource*) dan membandingkan performa algoritma Rijndael dan Twofish .

Pada tahapan ini juga dilakukan *benchmark* teknologi kedua algoritma untuk setiap mode kriptografi yang meliputi ECB, CBC, CFB dan OFB serta CTS dengan tujuan untuk mengetahui kebutuhan *resource memory* ataupun *procecessor*.

3.5 Kesimpulan dan Saran

Dalam tahap ini menentukan kesimpulan terhadap hasil analisa dan pengujian yang telah dilakukan. Hal ini untuk mengetahui apakah analisa dan pengujian dilakukan dapat membantu memberikan gambaran dan saran bagi praktisi kriptografer ataupun mayasrakat yang membutuhkan untuk kemananan data.

BAB IV
ANALISIS DAN PENGUJIAN

4.1 Parameter Rijndael dan Twofish

Pada BAB II dijelaskan bahwa Rijndael menggunakan jaringan substitusi dan permutasi, Rijndael yang merupakan kriptografi *bloc* membag kuncidan *plaintext* menjadi *matrix* 4 x 4 selanjutnya *plaintext* tersebut disebut *state* pada proses awal plaintext di-XOR-kan dengan key operasi ini disebut sebagai AddRoundKey, Selanjutnya masuk operas dominan yang berulang meliputi *SubByte*, *ShiftRow*, *MixColumn* dan *AddRoundKey* untuk kunci 256 *bit* dilakukan sebanyak 13 pengulangan operasi dominan kemudian pada tahapan akhir dilakukan proses *SubByte*, *ShiftRow* dan *AddRoundKey*.

Twofish menggunakan jaringan Feistel lihat gambar 2.6 juga merupakan teknologi kriptografi *block* pada tahapan awal atau inisiasi plaintext yang telah di bagi menjadi block-block data di-XOR-kan dengan kunci proses ini disebut dengan *Input Whitening*, selanjut pada proses utama yang meliputi Fungsi F, yang meliputi fungsi g, substitusi S-BOX, Proses MDS, PHT dan bit rotasi. Berikut adalah perbandingan parameter yang diperoleh dari implementasi kedua algoritma, parameter tersebut adalah :

Tabel 4.1 Parameter perbandingan algoritma

Pameter	Rijndael	Twofish
Jaringan	Substitusi	Feistel
R (Jumlah Round Enc/Dec)	14	16
Jumlah *Round Key*	R + 1	2R + 8
Ukuran *Block* Data (*bit*)	128	128
Penggunaaan Fungsi	SB, SR, MC, S-BOX, ARK, SB^{-1}, SR^{-1}, MC^{-1}	S-BOX, PHT, MDS
Penggunaan Operasi	<<, >>, $\oplus$	<<, >>, $\oplus$, +

Dari tabel 4.1 diketahui untuk panjang kunci 256 Rijndaelmembutuhkan 14 *round* dan 16 *round* untuk Twofish, untuk pembangunan kunci Rijndael hanya membutuhkan R + 1 sedangkan Twofish membutuhkan lebih dari 2 kali dari Rijndael yakni 2R + 8, untuk penggunaan fungsi Rinjdael menggunakan lebih banyak dibanding Twofish yakni meliputi *SubByte*, *ShiftRow, MixColoumn, S-BOX*, *AddRoundKey* dan *InverseSubByte*, *InverseShiftRow* Dan *MixColoumn* untuk proses dekripsi sedangkan Twofish menggunanakan S-BOX, PHT dan MDS *Matrix*.

4.2 Arsitektur Block Rijndael dan Twofish

Pada implementasi *input* ataupun *hardware* kriptografer harus memperhatikan arsitektur algoritma, karena arsitektur ini sangat mempengaruhi kinerja dan performance *input*. *XOR, mod* 2^{32}*Add* , *fixedshift*, GF(2^8) *multiply* dan *LUT* membutuhkan performance kecepatan kinerja *processor*.

Pada arsitektur Rijndael, S-box merupakan operasi dominan (*SubByte* dan *key schedule*) yang telah di deklarasikan sebelumnya, operasi ini membutuhkan 16 *copy* (salinan) tabel 8 x 8 S-box ini membutuhkan *resource* (*memory*), tetapi pada *byte swapping, galois field multipilation, key Addition* ditemukan struktur yang sangat simple dan ini hanya membutuhkan sedikit *resource*.

Tabel 4.2 Set Instruksi *Time Recovery* dan *Latency* pada Rijndael

Algoritma	Jumlah *Round*	Set Instruksi per-*Round*	Jumlah Set Instruksi	*Time Recovery (cycle)*	*Latency (cycle)*
Main *Round*	R=14	XOR(16)	224	224	224
		BXT(16)	224	224	224
		LOAD(20)	280	280	840

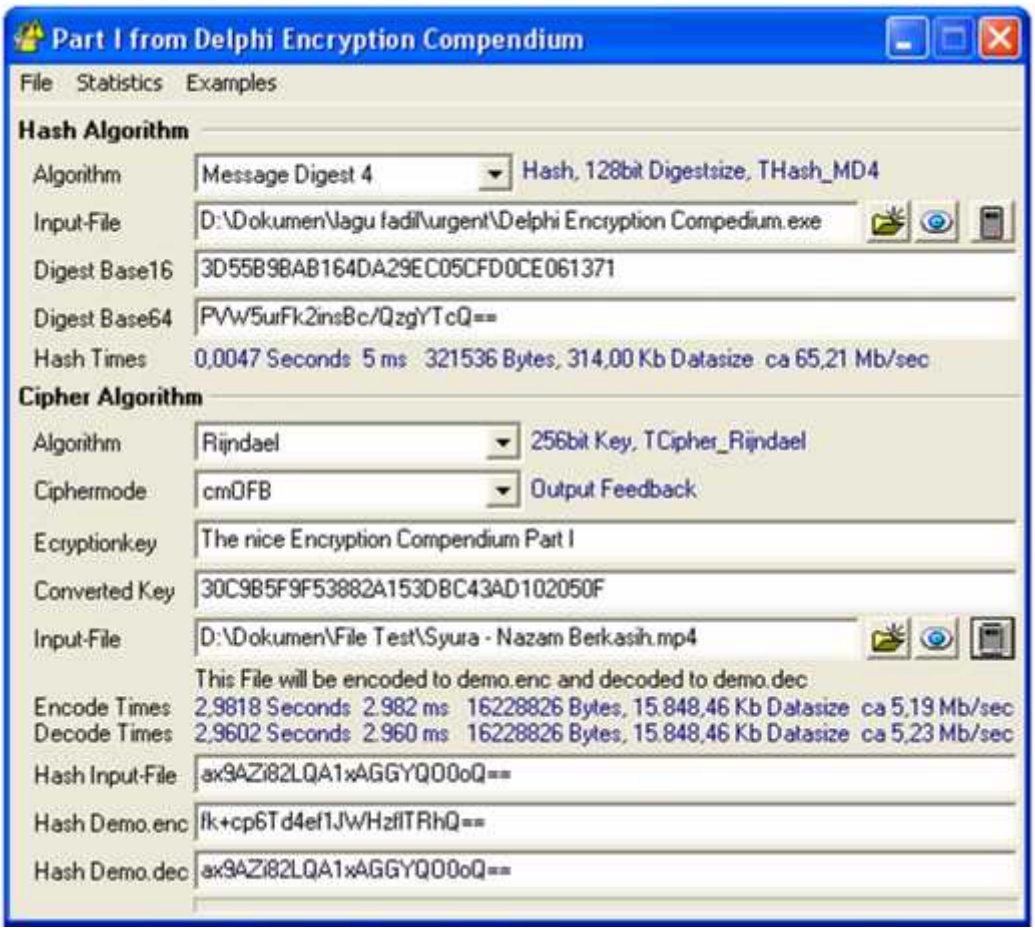
Final KeyAddition	1	XOR(4)	4	4	4
		LOAD(4)	4	4	12
Total			**736**	**736**	**1304**

Struktur Twofish juga mengunakan *S-box*, *S-box* di bangun oleh suatu fungsi yang meliputi Fungsi F, Fungsi g, S-Box, MDS *Matrix*, Transformasi PHT dan *Bit Rotation*, berikut detail arsitektur *block* yang membangun algoritma Twofish.

Tabel 4.3 Set Instruksi *Time Recovery* dan *Latency* pada Twofish

Algoritma	Jumlah *Round*	Set Instruksi per-*Round*	Jumlah Set Instruksi	*Time Recovery (cycle)*	*Latency (cycle)*
Main Round	R=16	ADD(4)	64	64	128
		XOR(8)	128	128	128
		ROT(2)	32	32	32
		BXT(8)	128	128	128
		LOAD(10)	160	160	480
Key Add ition Layer (in/out Whitening)	2R+8	XOR(4)	160	160	160
		LOAD(4)	160	160	480
Total			**832**	**832**	**1472**

4.3 Perbandingan Performa enkripsi dan dekripsi



4. 1 Pengujian Rijndael mode OFB pada file format MP4

Untuk pengujian performa waktu enkripsi dan dekripsi dilakukan menggunakan *software* uji Delphi Encryption Compedium dan *file* uji yang terdapat pada 4. 1. Berikut tampilan pengujian proses enkripsi dan dekripsi menggunakan algoritma Rijndael dengan mode OFB, dengan *input file* :" D:/Dokumen/File Test/Syura – Nazam Berkasih.mp4", *file* setelah di enkripsi Demo.enc dan *file* Seteleh di dekripsi menjadi demo.dec

4.3.1 Performa waktu Enkripsi

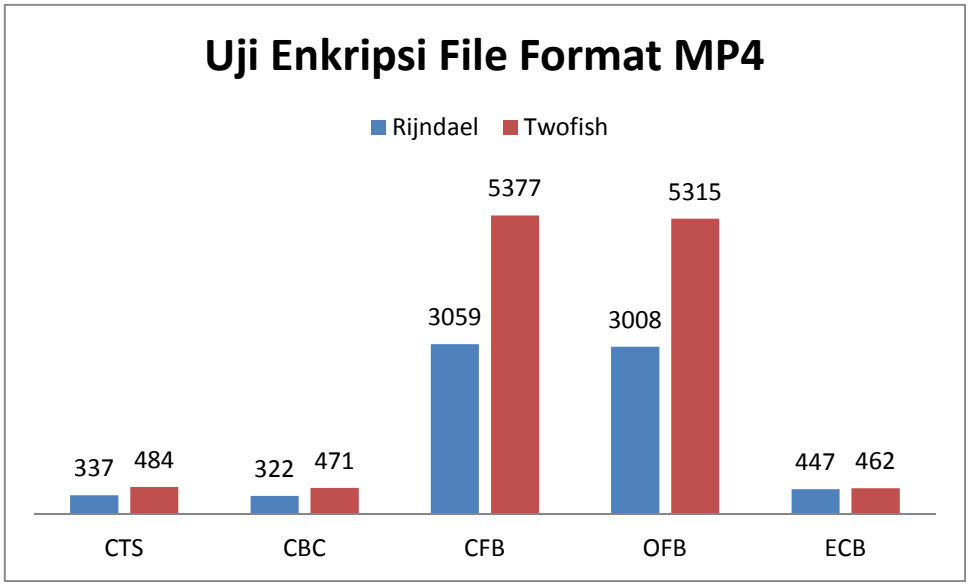
Tabel 4.4 Performa waktu enkripsi

Algoritma	Data	MODE (ms)				
		CTS	CBC	CFB	OFB	ECB
RIJNDAEL	TXT	1	1	1	1	1
	JPG	3	2	20	19	3
	PDF	19	13	118	115	13
	DOCX	24	24	203	197	54
	MP3	159	108	991	962	103
	MP4	337	322	3059	3008	447
TWOFISH	TXT	1	1	2	2	1
	JPG	4	4	34	34	4
	PDF	19	19	207	206	21
	DOCX	36	31	353	349	31
	MP3	154	152	1735	1730	148
	MP4	484	471	5377	5315	462

Dari tabel diatas di peroleh dari keseluruhan mode bahwa algoritma Rijndael membutuhkan waktu lebih sedikit untuk proses enkripsi daripada Twofish, kecualiuntuk file format PDF mode CTS waktu enkripsi Rijndael dan Twofish sama yakni CTS=”19”, sedangkan untuk format MP3 waktu enkripsi Rijndael CTS=”159” lebih besar dari pada Twofish yakni waktu CTS=”154” dan diketahui juga bahwa pada mode CFB dan OFB kedua algoritma menbutuhkan waktu yang besar.

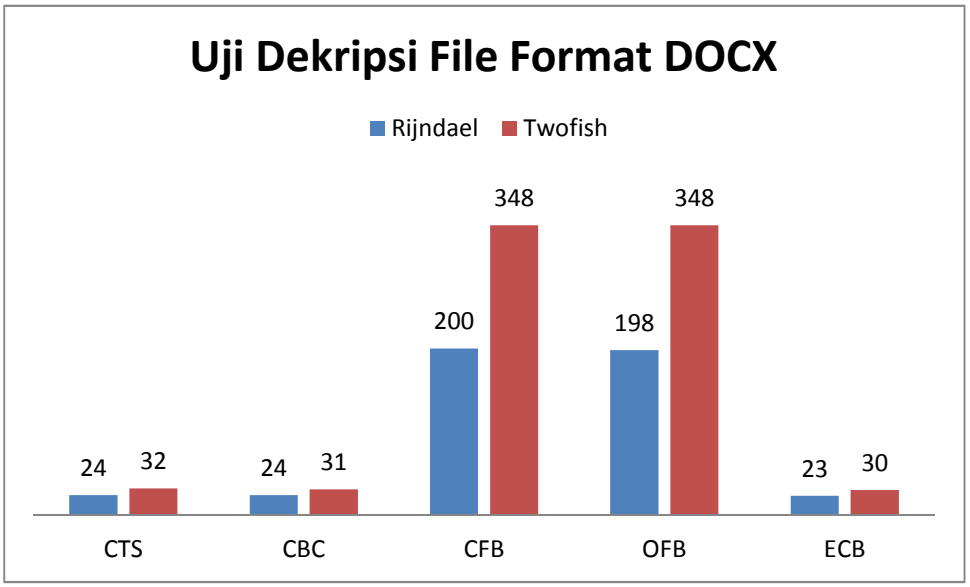
Grafik hasil uji enkripsi *file format* MP4 yang dijabarkan Tabel 4.4 untuk memudahkan perbandingan hasil uji enkripsi, grafik menunjukan untuk setiap

model kriptografi(CTS, CBC, CFB, OFB dan ECB) waktu yang digunakan Rijndael untuk proses enkripsi lebih kecil dari Twofish.



Gambar 4. 2 Enkripsi File Format MP4

4.3.2 Performa Waktu Dekripsi



Gambar 4.3Uji Dekripsi File Format DOCX

Pada grafik diatas waktu uji di peroleh Tabel 4.5 menggambarkan Rijndael mengkosumsi waktu lebih kecil dari Twofish untuk setiap model kriptografi

Tabel 4.5 Performa Waktu Dekripsi

Algoritma	Data	MODE (ms)				
		CTS	CBC	CFB	OFB	ECB
RIJNDAEL	TXT	0	0	1	1	1
	JPG	3	3	19	19	2
	PDF	14	14	116	116	13
	DOCX	24	24	200	198	23
	MP3	115	112	963	960	105
	MP4	565	370	2977	3002	771
TWOFISH	TXT	1	1	2	2	0
	JPG	4	4	34	34	3
	PDF	18	20	205	204	20
	DOCX	32	31	348	348	30
	MP3	149	154	1711	1729	146
	MP4	704	585	5294	5292	472

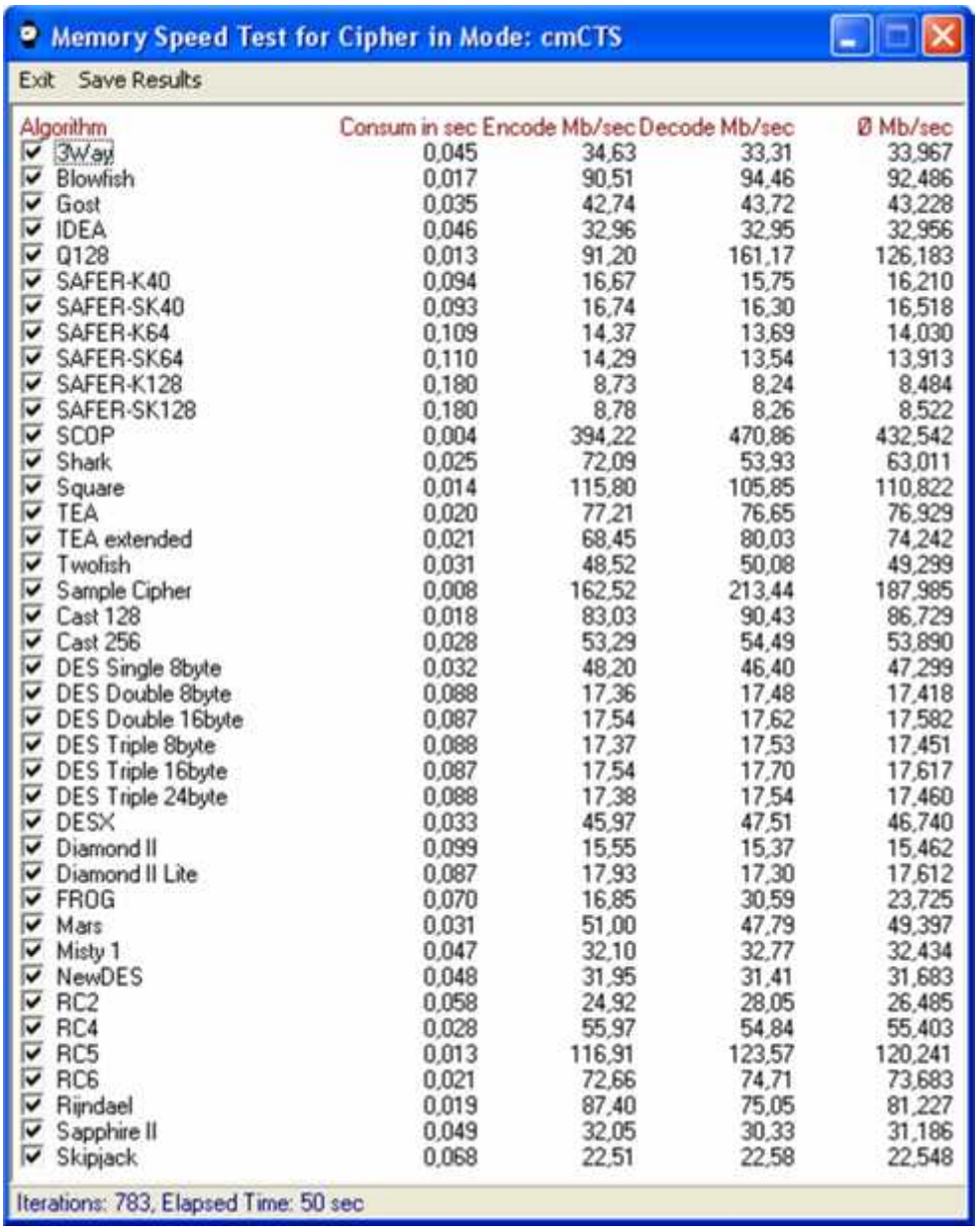
Dari tabel diatas di peroleh dari keseluruhan mode bahwa algoritma Rijndael membutuhkan waktu lebih sedikit untuk proses enkripsi daripada Twofish, dan diketahui juga bahwa pada mode CFB dan OFB kedua algoritma menbutuhkan waktu yang besar.

4.4 Performa Memory Speed

Uji performa *memory speed* dilakukan untuk mengetahui performa kinerja kedua algoritma pada Delphi Encryption Compedium setelah 50 detik program uji

memory speed dijalankan. Pengujian *memory speed* dilakukan untuk mengetahui jumlah iterasi, kecepatan memproses data pada enkripsi ataupun dekripsi

4.4.1 Performa Memory Speed Mode CTS



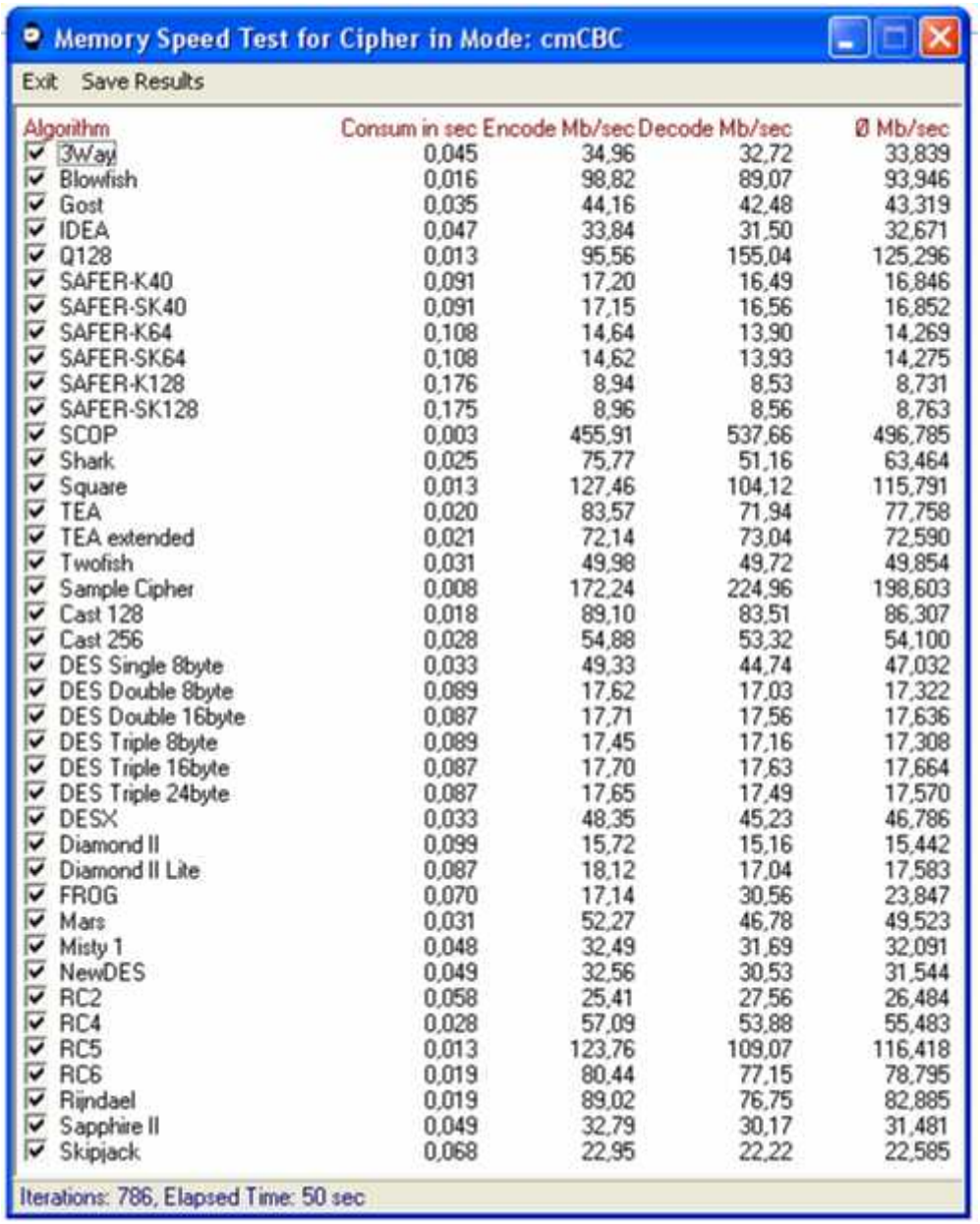
Algorithm	Consum in sec	Encode Mb/sec	Decode Mb/sec	Ø Mb/sec
3Way	0,045	34,63	33,31	33,967
Blowfish	0,017	90,51	94,46	92,486
Gost	0,035	42,74	43,72	43,228
IDEA	0,046	32,96	32,95	32,956
Q128	0,013	91,20	161,17	126,183
SAFER-K40	0,094	16,67	15,75	16,210
SAFER-SK40	0,093	16,74	16,30	16,518
SAFER-K64	0,109	14,37	13,69	14,030
SAFER-SK64	0,110	14,29	13,54	13,913
SAFER-K128	0,180	8,73	8,24	8,484
SAFER-SK128	0,180	8,78	8,26	8,522
SCOP	0,004	394,22	470,86	432,542
Shark	0,025	72,09	53,93	63,011
Square	0,014	115,80	105,85	110,822
TEA	0,020	77,21	76,65	76,929
TEA extended	0,021	68,45	80,03	74,242
Twofish	0,031	48,52	50,08	49,299
Sample Cipher	0,008	162,52	213,44	187,985
Cast 128	0,018	83,03	90,43	86,729
Cast 256	0,028	53,29	54,49	53,890
DES Single 8byte	0,032	48,20	46,40	47,299
DES Double 8byte	0,088	17,36	17,48	17,418
DES Double 16byte	0,087	17,54	17,62	17,582
DES Triple 8byte	0,088	17,37	17,53	17,451
DES Triple 16byte	0,087	17,54	17,70	17,617
DES Triple 24byte	0,088	17,38	17,54	17,460
DESX	0,033	45,97	47,51	46,740
Diamond II	0,099	15,55	15,37	15,462
Diamond II Lite	0,087	17,93	17,30	17,612
FROG	0,070	16,85	30,59	23,725
Mars	0,031	51,00	47,79	49,397
Misty 1	0,047	32,10	32,77	32,434
NewDES	0,048	31,95	31,41	31,683
RC2	0,058	24,92	28,05	26,485
RC4	0,028	55,97	54,84	55,403
RC5	0,013	116,91	123,57	120,241
RC6	0,021	72,66	74,71	73,683
Rijndael	0,019	87,40	75,05	81,227
Sapphire II	0,049	32,05	30,33	31,186
Skipjack	0,068	22,51	22,58	22,548

Gambar 4.4 Performa Memory Speed mode CTS

Dari Gambar 4.4 Pengujian *memory speed* melakukan 783 iterasi, proses enkripsi Rijndael 87.40 Mb/detik dekripsi 75.05 Mb/detik dan membutuhkan waktu 0.019

detik, sedangkan Proses enkripsi Twofish 48.52 Mb/detik dekripsi 50.08 Mb/detik dan membutuhkan 0.031 detik.

4.4.2 Performa Memory Speed Mode CBC



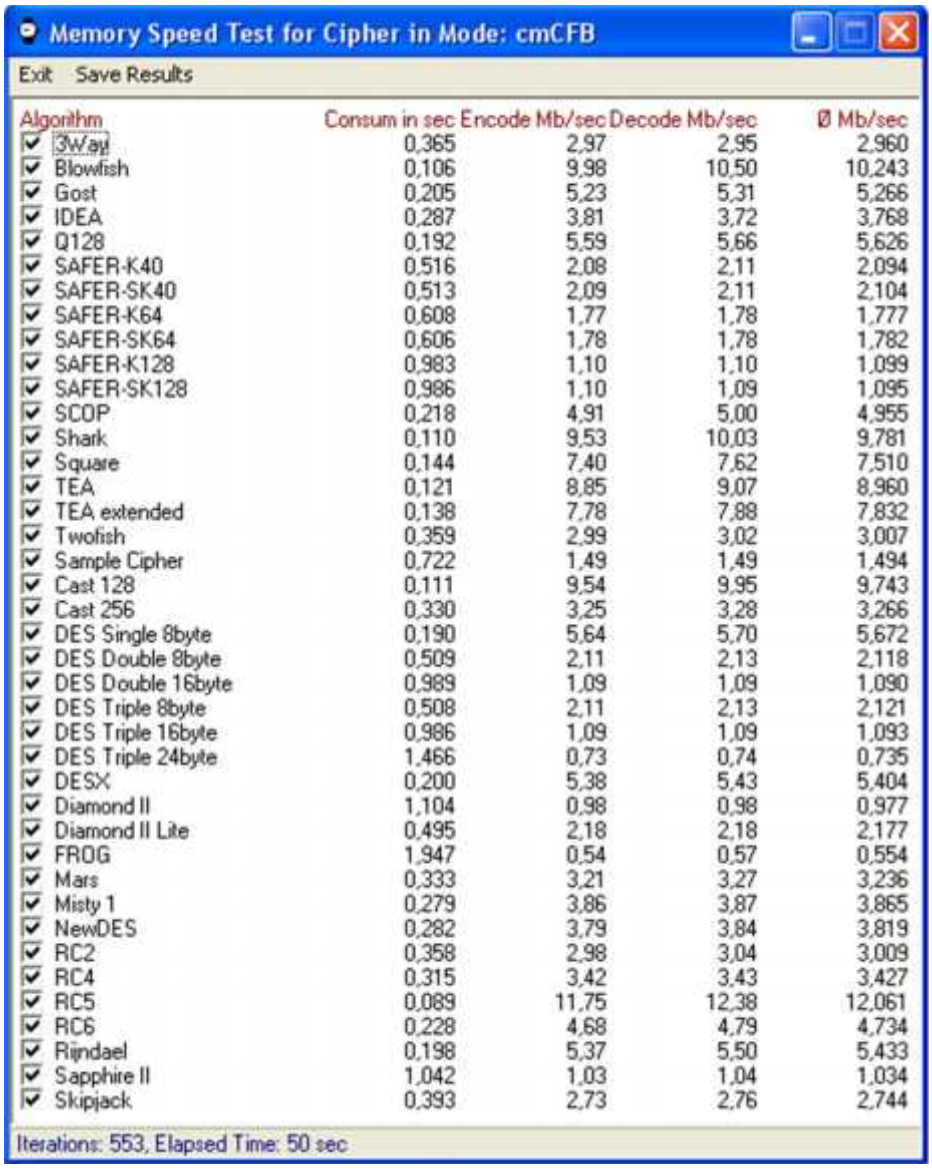
Algorithm	Consum in sec	Encode Mb/sec	Decode Mb/sec	Ø Mb/sec
3Way	0,045	34,96	32,72	33,839
Blowfish	0,016	98,82	89,07	93,946
Gost	0,035	44,16	42,48	43,319
IDEA	0,047	33,84	31,50	32,671
Q128	0,013	95,56	155,04	125,296
SAFER-K40	0,091	17,20	16,49	16,846
SAFER-SK40	0,091	17,15	16,56	16,852
SAFER-K64	0,108	14,64	13,90	14,269
SAFER-SK64	0,108	14,62	13,93	14,275
SAFER-K128	0,176	8,94	8,53	8,731
SAFER-SK128	0,175	8,96	8,56	8,763
SCOP	0,003	455,91	537,66	496,785
Shark	0,025	75,77	51,16	63,464
Square	0,013	127,46	104,12	115,791
TEA	0,020	83,57	71,94	77,758
TEA extended	0,021	72,14	73,04	72,590
Twofish	0,031	49,98	49,72	49,854
Sample Cipher	0,008	172,24	224,96	198,603
Cast 128	0,018	89,10	83,51	86,307
Cast 256	0,028	54,88	53,32	54,100
DES Single 8byte	0,033	49,33	44,74	47,032
DES Double 8byte	0,089	17,62	17,03	17,322
DES Double 16byte	0,087	17,71	17,56	17,636
DES Triple 8byte	0,089	17,45	17,16	17,308
DES Triple 16byte	0,087	17,70	17,63	17,664
DES Triple 24byte	0,087	17,65	17,49	17,570
DESX	0,033	48,35	45,23	46,786
Diamond II	0,099	15,72	15,16	15,442
Diamond II Lite	0,087	18,12	17,04	17,583
FROG	0,070	17,14	30,56	23,847
Mars	0,031	52,27	46,78	49,523
Misty 1	0,048	32,49	31,69	32,091
NewDES	0,049	32,56	30,53	31,544
RC2	0,058	25,41	27,56	26,484
RC4	0,028	57,09	53,88	55,483
RC5	0,013	123,76	109,07	116,418
RC6	0,019	80,44	77,15	78,795
Rijndael	0,019	89,02	76,75	82,885
Sapphire II	0,049	32,79	30,17	31,481
Skipjack	0,068	22,95	22,22	22,585

Gambar 4.5 Performa Memory Speed mode CBC

Dari Gambar 4.5 Pengujian *memory speed* melakukan 786 iterasi, proses enkripsi Rijndael 89.02 Mb/detik dekripsi 76.75 Mb/detik dan membutuhkan waktu 0.019

detik, sedangkan Proses enkripsi Twofish 49.98 Mb/detik dekripsi 49.72 Mb/detik dan membutuhkan 0.031 detik.

4.4.3 Performa Memory Speed Mode CFB



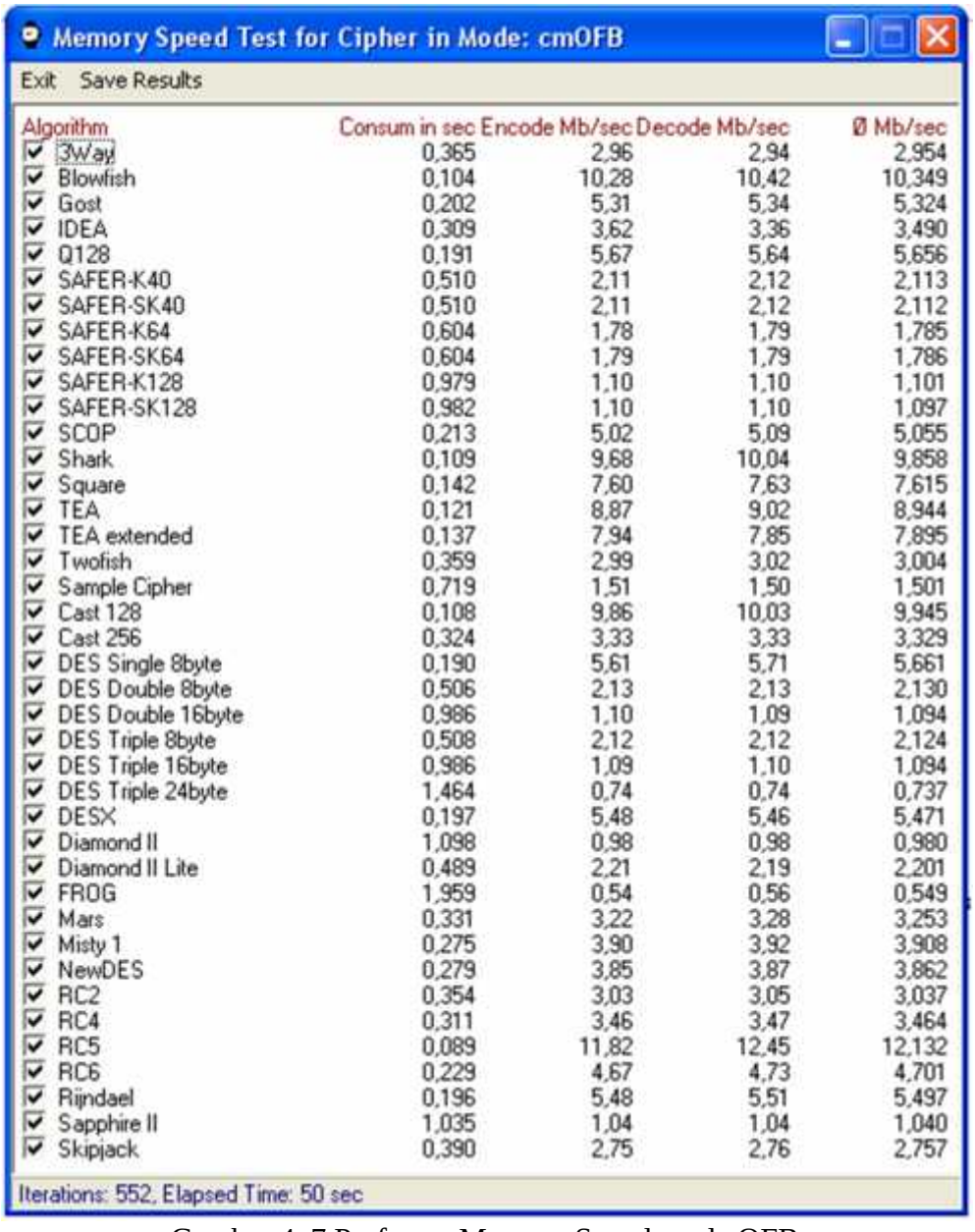
Algorithm	Consum in sec	Encode Mb/sec	Decode Mb/sec	Ø Mb/sec
3Way	0,365	2,97	2,95	2,960
Blowfish	0,106	9,98	10,50	10,243
Gost	0,205	5,23	5,31	5,266
IDEA	0,287	3,81	3,72	3,768
Q128	0,192	5,59	5,66	5,626
SAFER-K40	0,516	2,08	2,11	2,094
SAFER-SK40	0,513	2,09	2,11	2,104
SAFER-K64	0,608	1,77	1,78	1,777
SAFER-SK64	0,606	1,78	1,78	1,782
SAFER-K128	0,983	1,10	1,10	1,099
SAFER-SK128	0,986	1,10	1,09	1,095
SCOP	0,218	4,91	5,00	4,955
Shark	0,110	9,53	10,03	9,781
Square	0,144	7,40	7,62	7,510
TEA	0,121	8,85	9,07	8,960
TEA extended	0,138	7,78	7,88	7,832
Twofish	0,359	2,99	3,02	3,007
Sample Cipher	0,722	1,49	1,49	1,494
Cast 128	0,111	9,54	9,95	9,743
Cast 256	0,330	3,25	3,28	3,266
DES Single 8byte	0,190	5,64	5,70	5,672
DES Double 8byte	0,509	2,11	2,13	2,118
DES Double 16byte	0,989	1,09	1,09	1,090
DES Triple 8byte	0,508	2,11	2,13	2,121
DES Triple 16byte	0,986	1,09	1,09	1,093
DES Triple 24byte	1,466	0,73	0,74	0,735
DESX	0,200	5,38	5,43	5,404
Diamond II	1,104	0,98	0,98	0,977
Diamond II Lite	0,495	2,18	2,18	2,177
FROG	1,947	0,54	0,57	0,554
Mars	0,333	3,21	3,27	3,236
Misty 1	0,279	3,86	3,87	3,865
NewDES	0,282	3,79	3,84	3,819
RC2	0,358	2,98	3,04	3,009
RC4	0,315	3,42	3,43	3,427
RC5	0,089	11,75	12,38	12,061
RC6	0,228	4,68	4,79	4,734
Rijndael	0,198	5,37	5,50	5,433
Sapphire II	1,042	1,03	1,04	1,034
Skipjack	0,393	2,73	2,76	2,744

Gambar 4.6 Performa Memory Speed mode CFB

Dari Gambar 4. 7 Pengujian *memory speed* melakukan 553 iterasi, proses enkripsi Rijndael 4.68 Mb/detik dekripsi 4.79 Mb/detik dan membutuhkan waktu 0.196 detik, sedangkan Proses enkripsi Twofish 2.99 Mb/detik dekripsi 3.02 Mb/detik dan membutuhkan 0.359 detik.

4.4.4 Performa Memory Speed Mode OFB

Dari Gambar 4. 7 Pengujian *memory speed* melakukan 552 iterasi, proses enkripsi Rijndael 5.48 Mb/detik dekripsi 5.51 Mb/detik dan membutuhkan waktu 0.196 detik, sedangkan Proses enkripsi Twofish 2.99 Mb/detik dekripsi 3.02 Mb/detik dan membutuhkan 0.359 detik.

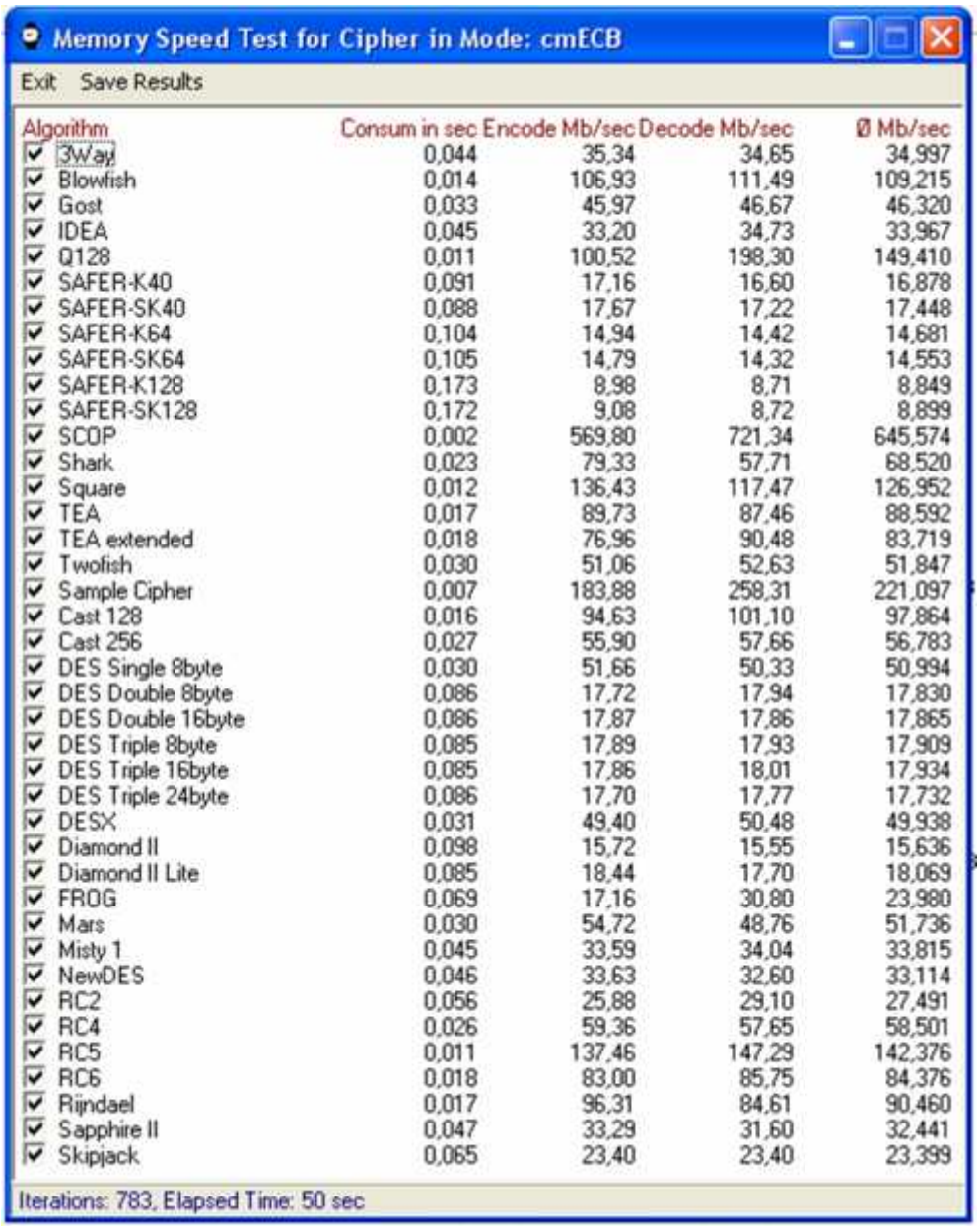


Algorithm	Consum in sec	Encode Mb/sec	Decode Mb/sec	Ø Mb/sec
3Way	0,365	2,96	2,94	2,954
Blowfish	0,104	10,28	10,42	10,349
Gost	0,202	5,31	5,34	5,324
IDEA	0,309	3,62	3,36	3,490
Q128	0,191	5,67	5,64	5,656
SAFER-K40	0,510	2,11	2,12	2,113
SAFER-SK40	0,510	2,11	2,12	2,112
SAFER-K64	0,604	1,78	1,79	1,785
SAFER-SK64	0,604	1,79	1,79	1,786
SAFER-K128	0,979	1,10	1,10	1,101
SAFER-SK128	0,982	1,10	1,10	1,097
SCOP	0,213	5,02	5,09	5,055
Shark	0,109	9,68	10,04	9,858
Square	0,142	7,60	7,63	7,615
TEA	0,121	8,87	9,02	8,944
TEA extended	0,137	7,94	7,85	7,895
Twofish	0,359	2,99	3,02	3,004
Sample Cipher	0,719	1,51	1,50	1,501
Cast 128	0,108	9,86	10,03	9,945
Cast 256	0,324	3,33	3,33	3,329
DES Single 8byte	0,190	5,61	5,71	5,661
DES Double 8byte	0,506	2,13	2,13	2,130
DES Double 16byte	0,986	1,10	1,09	1,094
DES Triple 8byte	0,508	2,12	2,12	2,124
DES Triple 16byte	0,986	1,09	1,10	1,094
DES Triple 24byte	1,464	0,74	0,74	0,737
DESX	0,197	5,48	5,46	5,471
Diamond II	1,098	0,98	0,98	0,980
Diamond II Lite	0,489	2,21	2,19	2,201
FROG	1,959	0,54	0,56	0,549
Mars	0,331	3,22	3,28	3,253
Misty 1	0,275	3,90	3,92	3,908
NewDES	0,279	3,85	3,87	3,862
RC2	0,354	3,03	3,05	3,037
RC4	0,311	3,46	3,47	3,464
RC5	0,089	11,82	12,45	12,132
RC6	0,229	4,67	4,73	4,701
Rijndael	0,196	5,48	5,51	5,497
Sapphire II	1,035	1,04	1,04	1,040
Skipjack	0,390	2,75	2,76	2,757

Gambar 4. 7 Performa Memory Speed mode OFB

4.4.5 Performa Memory Speed Mode ECB

Dari Gambar 4. 8 Pengujian *memory speed* melakukan 783 iterasi, proses enkripsi Rijndael 96.31 Mb/detik dekripsi 84.61 Mb/detik dan membutuhkan waktu 0.017 detik, sedangkan Proses enkripsi Twofish 51.06 Mb/detik dekripsi 52.63 Mb/detik dan membutuhkan 0.030 detik.



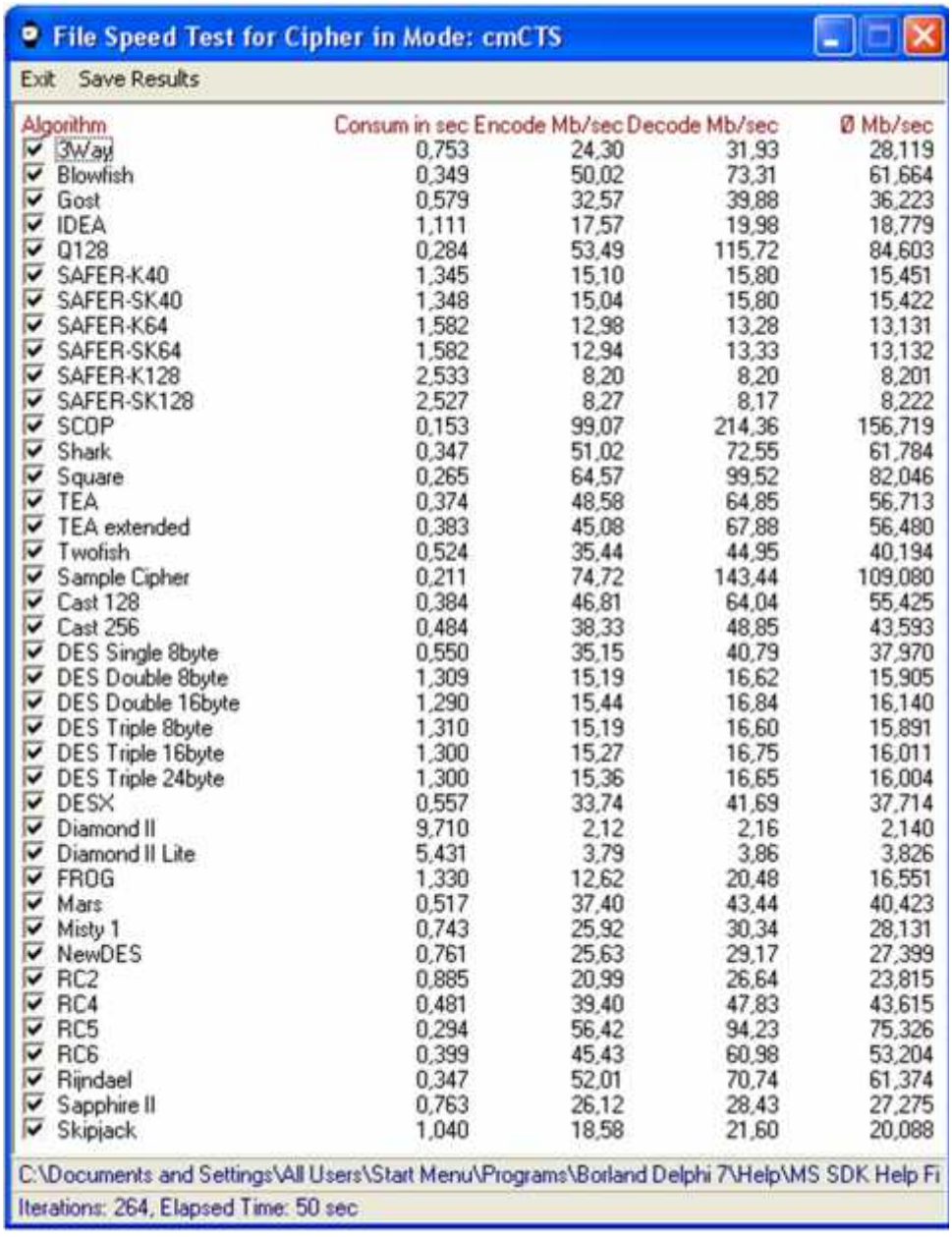
Algorithm	Consum in sec	Encode Mb/sec	Decode Mb/sec	Ø Mb/sec
3Way	0,044	35,34	34,65	34,997
Blowfish	0,014	106,93	111,49	109,215
Gost	0,033	45,97	46,67	46,320
IDEA	0,045	33,20	34,73	33,967
Q128	0,011	100,52	198,30	149,410
SAFER-K40	0,091	17,16	16,60	16,878
SAFER-SK40	0,088	17,67	17,22	17,448
SAFER-K64	0,104	14,94	14,42	14,681
SAFER-SK64	0,105	14,79	14,32	14,553
SAFER-K128	0,173	8,98	8,71	8,849
SAFER-SK128	0,172	9,08	8,72	8,899
SCOP	0,002	569,80	721,34	645,574
Shark	0,023	79,33	57,71	68,520
Square	0,012	136,43	117,47	126,952
TEA	0,017	89,73	87,46	88,592
TEA extended	0,018	76,96	90,48	83,719
Twofish	0,030	51,06	52,63	51,847
Sample Cipher	0,007	183,88	258,31	221,097
Cast 128	0,016	94,63	101,10	97,864
Cast 256	0,027	55,90	57,66	56,783
DES Single 8byte	0,030	51,66	50,33	50,994
DES Double 8byte	0,086	17,72	17,94	17,830
DES Double 16byte	0,086	17,87	17,86	17,865
DES Triple 8byte	0,085	17,89	17,93	17,909
DES Triple 16byte	0,085	17,86	18,01	17,934
DES Triple 24byte	0,086	17,70	17,77	17,732
DESX	0,031	49,40	50,48	49,938
Diamond II	0,098	15,72	15,55	15,636
Diamond II Lite	0,085	18,44	17,70	18,069
FROG	0,069	17,16	30,80	23,980
Mars	0,030	54,72	48,76	51,736
Misty 1	0,045	33,59	34,04	33,815
NewDES	0,046	33,63	32,60	33,114
RC2	0,056	25,88	29,10	27,491
RC4	0,026	59,36	57,65	58,501
RC5	0,011	137,46	147,29	142,376
RC6	0,018	83,00	85,75	84,376
Rijndael	0,017	96,31	84,61	90,460
Sapphire II	0,047	33,29	31,60	32,441
Skipjack	0,065	23,40	23,40	23,399

Gambar 4. 8 Performa Memory Speed mode ECB

4.5 Performa File Speed

Performa *File speed* adalah preforma iterasi yang ada pada DEC untuk mengetahui performa algoritma mengenkripsi dan dekripsi masukan setelah waktu yang ditentukan pada penelitian ini batas waktu ditentukan setelah proses berjalan selama 50 detik

4.5.1 Performa File Speed Mode CTS

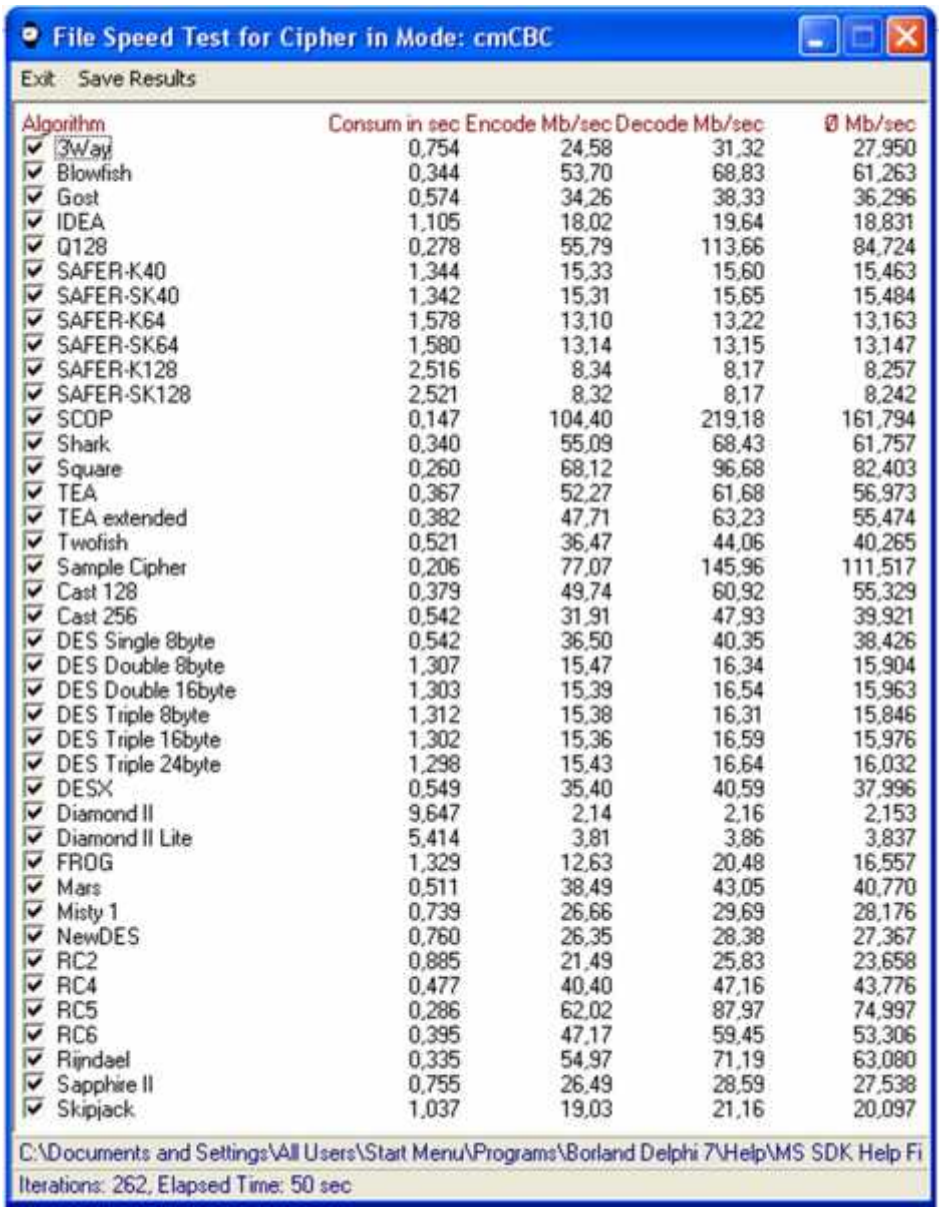


Algorithm	Consum in sec	Encode Mb/sec	Decode Mb/sec	Ø Mb/sec
3Way	0,753	24,30	31,93	28,119
Blowfish	0,349	50,02	73,31	61,664
Gost	0,579	32,57	39,88	36,223
IDEA	1,111	17,57	19,98	18,779
Q128	0,284	53,49	115,72	84,603
SAFER-K40	1,345	15,10	15,80	15,451
SAFER-SK40	1,348	15,04	15,80	15,422
SAFER-K64	1,582	12,98	13,28	13,131
SAFER-SK64	1,582	12,94	13,33	13,132
SAFER-K128	2,533	8,20	8,20	8,201
SAFER-SK128	2,527	8,27	8,17	8,222
SCOP	0,153	99,07	214,36	156,719
Shark	0,347	51,02	72,55	61,784
Square	0,265	64,57	99,52	82,046
TEA	0,374	48,58	64,85	56,713
TEA extended	0,383	45,08	67,88	56,480
Twofish	0,524	35,44	44,95	40,194
Sample Cipher	0,211	74,72	143,44	109,080
Cast 128	0,384	46,81	64,04	55,425
Cast 256	0,484	38,33	48,85	43,593
DES Single 8byte	0,550	35,15	40,79	37,970
DES Double 8byte	1,309	15,19	16,62	15,905
DES Double 16byte	1,290	15,44	16,84	16,140
DES Triple 8byte	1,310	15,19	16,60	15,891
DES Triple 16byte	1,300	15,27	16,75	16,011
DES Triple 24byte	1,300	15,36	16,65	16,004
DESX	0,557	33,74	41,69	37,714
Diamond II	9,710	2,12	2,16	2,140
Diamond II Lite	5,431	3,79	3,86	3,826
FROG	1,330	12,62	20,48	16,551
Mars	0,517	37,40	43,44	40,423
Misty 1	0,743	25,92	30,34	28,131
NewDES	0,761	25,63	29,17	27,399
RC2	0,885	20,99	26,64	23,815
RC4	0,481	39,40	47,83	43,615
RC5	0,294	56,42	94,23	75,326
RC6	0,399	45,43	60,98	53,204
Rijndael	0,347	52,01	70,74	61,374
Sapphire II	0,763	26,12	28,43	27,275
Skipjack	1,040	18,58	21,60	20,088

Gambar 4. 9 Performa File Speed mode CTS

Dari Gambar 4. 9 Pengujian *File speed* melakukan 264 iterasi, proses enkripsi Rijndael 52.01 Mb/detik dekripsi 70.74 Mb/detik dan membutuhkan waktu 0.347 detik, sedangkan Proses enkripsi Twofish 35.44 Mb/detik dekripsi 44.95 Mb/detik dan membutuhkan 0.524 detik.

4.5.2 Performa File Speed Mode CBC



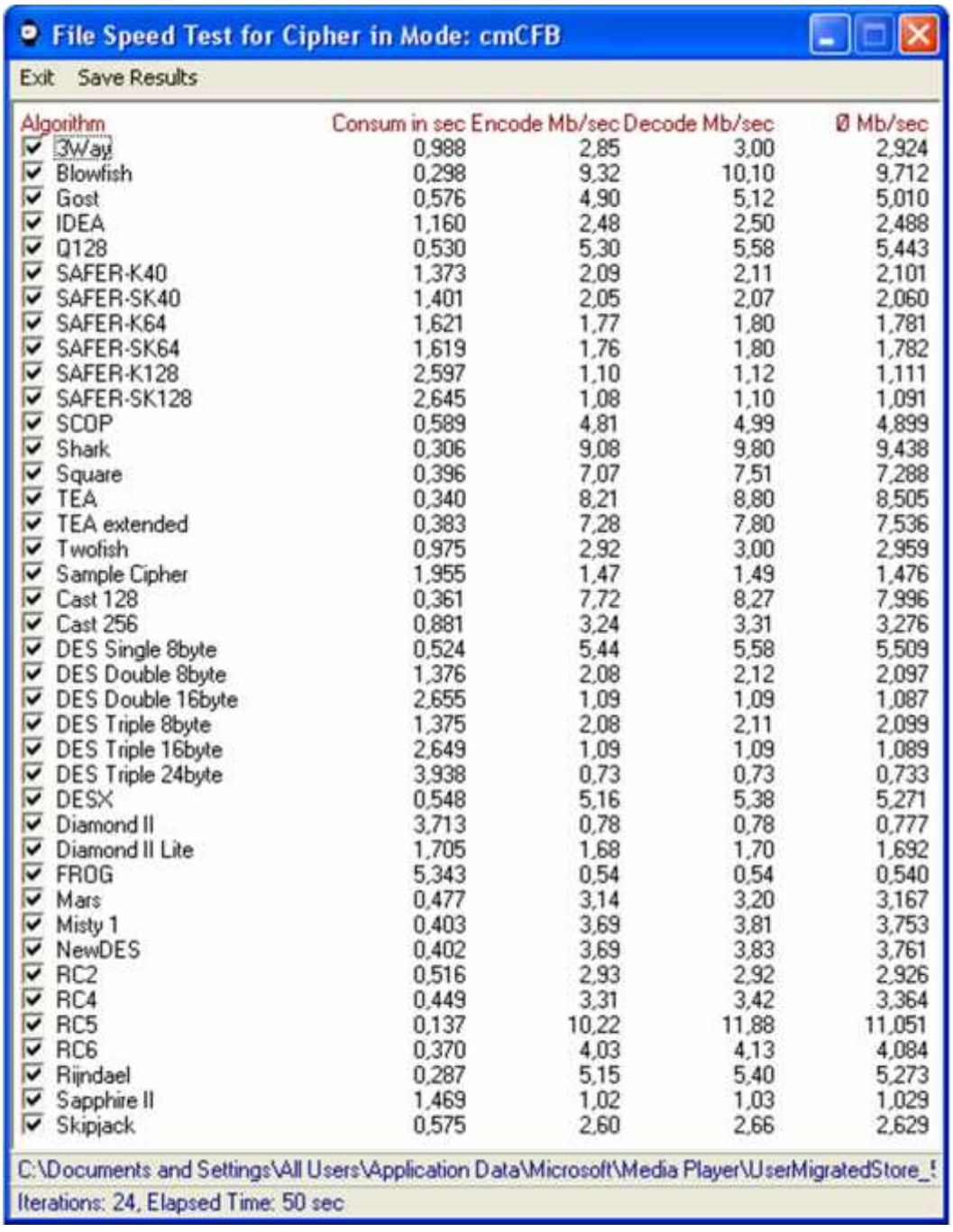
Algorithm	Consum in sec	Encode Mb/sec	Decode Mb/sec	Ø Mb/sec
3Way	0,754	24,58	31,32	27,950
Blowfish	0,344	53,70	68,83	61,263
Gost	0,574	34,26	38,33	36,296
IDEA	1,105	18,02	19,64	18,831
Q128	0,278	55,79	113,66	84,724
SAFER-K40	1,344	15,33	15,60	15,463
SAFER-SK40	1,342	15,31	15,65	15,484
SAFER-K64	1,578	13,10	13,22	13,163
SAFER-SK64	1,580	13,14	13,15	13,147
SAFER-K128	2,516	8,34	8,17	8,257
SAFER-SK128	2,521	8,32	8,17	8,242
SCOP	0,147	104,40	219,18	161,794
Shark	0,340	55,09	68,43	61,757
Square	0,260	68,12	96,68	82,403
TEA	0,367	52,27	61,68	56,973
TEA extended	0,382	47,71	63,23	55,474
Twofish	0,521	36,47	44,06	40,265
Sample Cipher	0,206	77,07	145,96	111,517
Cast 128	0,379	49,74	60,92	55,329
Cast 256	0,542	31,91	47,93	39,921
DES Single 8byte	0,542	36,50	40,35	38,426
DES Double 8byte	1,307	15,47	16,34	15,904
DES Double 16byte	1,303	15,39	16,54	15,963
DES Triple 8byte	1,312	15,38	16,31	15,846
DES Triple 16byte	1,302	15,36	16,59	15,976
DES Triple 24byte	1,298	15,43	16,64	16,032
DESX	0,549	35,40	40,59	37,996
Diamond II	9,647	2,14	2,16	2,153
Diamond II Lite	5,414	3,81	3,86	3,837
FROG	1,329	12,63	20,48	16,557
Mars	0,511	38,49	43,05	40,770
Misty 1	0,739	26,66	29,69	28,176
NewDES	0,760	26,35	28,38	27,367
RC2	0,885	21,49	25,83	23,658
RC4	0,477	40,40	47,16	43,776
RC5	0,286	62,02	87,97	74,997
RC6	0,395	47,17	59,45	53,306
Rijndael	0,335	54,97	71,19	63,080
Sapphire II	0,755	26,49	28,59	27,538
Skipjack	1,037	19,03	21,16	20,097

Gambar 4.10 Performa File Speed mode CBC

Dari Gambar 4.10 Pengujian *File speed* melakukan 262 iterasi, proses enkripsi Rijndael 54.97 Mb/detik dekripsi 71.19 Mb/detik dan membutuhkan

waktu 0.335 detik, sedangkan Proses enkripsi Twofish 36.47 Mb/detik dekripsi 44.06 Mb/detik dan membutuhkan 0.521 detik.

4.5.3 Performa File Speed Mode CFB

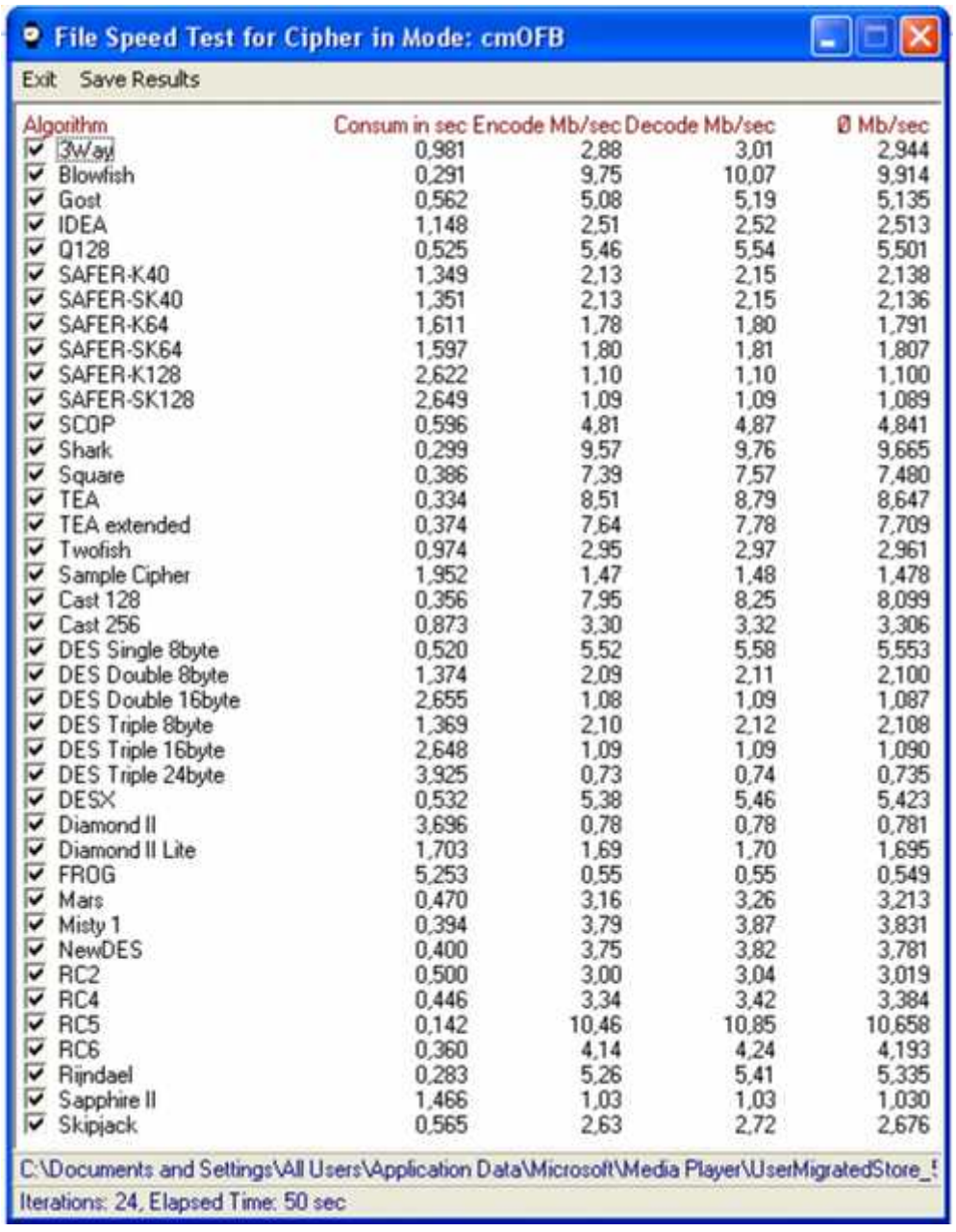


Algorithm	Consum in sec	Encode Mb/sec	Decode Mb/sec	Ø Mb/sec
3Way	0,988	2,85	3,00	2,924
Blowfish	0,298	9,32	10,10	9,712
Gost	0,576	4,90	5,12	5,010
IDEA	1,160	2,48	2,50	2,488
Q128	0,530	5,30	5,58	5,443
SAFER-K40	1,373	2,09	2,11	2,101
SAFER-SK40	1,401	2,05	2,07	2,060
SAFER-K64	1,621	1,77	1,80	1,781
SAFER-SK64	1,619	1,76	1,80	1,782
SAFER-K128	2,597	1,10	1,12	1,111
SAFER-SK128	2,645	1,08	1,10	1,091
SCOP	0,589	4,81	4,99	4,899
Shark	0,306	9,08	9,80	9,438
Square	0,396	7,07	7,51	7,288
TEA	0,340	8,21	8,80	8,505
TEA extended	0,383	7,28	7,80	7,536
Twofish	0,975	2,92	3,00	2,959
Sample Cipher	1,955	1,47	1,49	1,476
Cast 128	0,361	7,72	8,27	7,996
Cast 256	0,881	3,24	3,31	3,276
DES Single 8byte	0,524	5,44	5,58	5,509
DES Double 8byte	1,376	2,08	2,12	2,097
DES Double 16byte	2,655	1,09	1,09	1,087
DES Triple 8byte	1,375	2,08	2,11	2,099
DES Triple 16byte	2,649	1,09	1,09	1,089
DES Triple 24byte	3,938	0,73	0,73	0,733
DESX	0,548	5,16	5,38	5,271
Diamond II	3,713	0,78	0,78	0,777
Diamond II Lite	1,705	1,68	1,70	1,692
FROG	5,343	0,54	0,54	0,540
Mars	0,477	3,14	3,20	3,167
Misty 1	0,403	3,69	3,81	3,753
NewDES	0,402	3,69	3,83	3,761
RC2	0,516	2,93	2,92	2,926
RC4	0,449	3,31	3,42	3,364
RC5	0,137	10,22	11,88	11,051
RC6	0,370	4,03	4,13	4,084
Rijndael	0,287	5,15	5,40	5,273
Sapphire II	1,469	1,02	1,03	1,029
Skipjack	0,575	2,60	2,66	2,629

Gambar 4.11 Performa File Speed mode CFB

Dari Gambar 4.11uji *File speed* melakukan 24 iterasi, proses enkripsi Rijndael 5.15 Mb/detik dekripsi 5.40 Mb/detik dan membutuhkan waktu 0.287 detik, sedangkan Proses enkripsi Twofish 2.92 Mb/detik dekripsi 3.00 Mb/detik dan membutuhkan 0.975 detik.

4.5.4 Performa File Speed Mode OFB

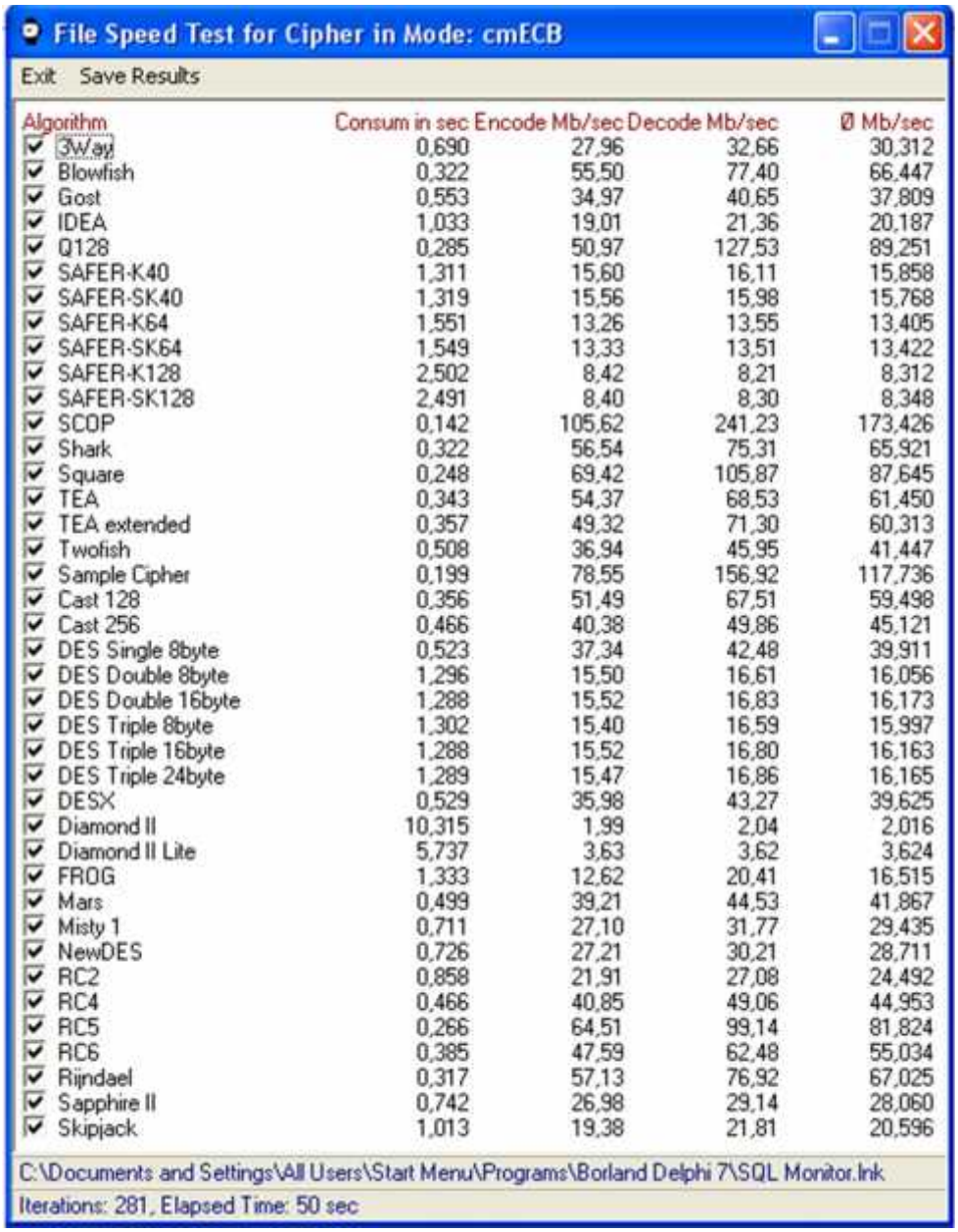


Algorithm	Consum in sec	Encode Mb/sec	Decode Mb/sec	Ø Mb/sec
3Way	0,981	2,88	3,01	2,944
Blowfish	0,291	9,75	10,07	9,914
Gost	0,562	5,08	5,19	5,135
IDEA	1,148	2,51	2,52	2,513
Q128	0,525	5,46	5,54	5,501
SAFER-K40	1,349	2,13	2,15	2,138
SAFER-SK40	1,351	2,13	2,15	2,136
SAFER-K64	1,611	1,78	1,80	1,791
SAFER-SK64	1,597	1,80	1,81	1,807
SAFER-K128	2,622	1,10	1,10	1,100
SAFER-SK128	2,649	1,09	1,09	1,089
SCOP	0,596	4,81	4,87	4,841
Shark	0,299	9,57	9,76	9,665
Square	0,386	7,39	7,57	7,480
TEA	0,334	8,51	8,79	8,647
TEA extended	0,374	7,64	7,78	7,709
Twofish	0,974	2,95	2,97	2,961
Sample Cipher	1,952	1,47	1,48	1,478
Cast 128	0,356	7,95	8,25	8,099
Cast 256	0,873	3,30	3,32	3,306
DES Single 8byte	0,520	5,52	5,58	5,553
DES Double 8byte	1,374	2,09	2,11	2,100
DES Double 16byte	2,655	1,08	1,09	1,087
DES Triple 8byte	1,369	2,10	2,12	2,108
DES Triple 16byte	2,648	1,09	1,09	1,090
DES Triple 24byte	3,925	0,73	0,74	0,735
DESX	0,532	5,38	5,46	5,423
Diamond II	3,696	0,78	0,78	0,781
Diamond II Lite	1,703	1,69	1,70	1,695
FROG	5,253	0,55	0,55	0,549
Mars	0,470	3,16	3,26	3,213
Misty 1	0,394	3,79	3,87	3,831
NewDES	0,400	3,75	3,82	3,781
RC2	0,500	3,00	3,04	3,019
RC4	0,446	3,34	3,42	3,384
RC5	0,142	10,46	10,85	10,658
RC6	0,360	4,14	4,24	4,193
Rijndael	0,283	5,26	5,41	5,335
Sapphire II	1,466	1,03	1,03	1,030
Skipjack	0,565	2,63	2,72	2,676

Gambar 4.12 Performa File Speed mode OFB

Dari Gambar 4.12 Pengujian *File speed* melakukan 24 iterasi, proses enkripsi Rijndael 5.26 Mb/detik dekripsi 5.41 Mb/detik dan membutuhkan waktu 0.283 detik, sedangkan Proses enkripsi Twofish 2.95 Mb/detik dekripsi 2.97 Mb/detik dan membutuhkan 0.974 detik.

4.5.5 Performa File Speed Mode ECB



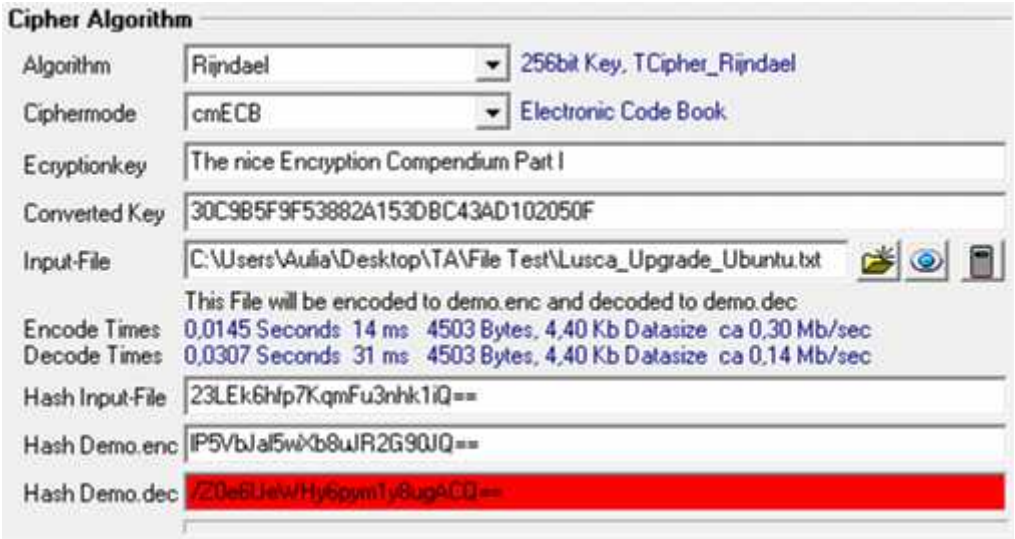
Algorithm	Consum in sec	Encode Mb/sec	Decode Mb/sec	Ø Mb/sec
3Way	0,690	27,96	32,66	30,312
Blowfish	0,322	55,50	77,40	66,447
Gost	0,553	34,97	40,65	37,809
IDEA	1,033	19,01	21,36	20,187
Q128	0,285	50,97	127,53	89,251
SAFER-K40	1,311	15,60	16,11	15,858
SAFER-SK40	1,319	15,56	15,98	15,768
SAFER-K64	1,551	13,26	13,55	13,405
SAFER-SK64	1,549	13,33	13,51	13,422
SAFER-K128	2,502	8,42	8,21	8,312
SAFER-SK128	2,491	8,40	8,30	8,348
SCOP	0,142	105,62	241,23	173,426
Shark	0,322	56,54	75,31	65,921
Square	0,248	69,42	105,87	87,645
TEA	0,343	54,37	68,53	61,450
TEA extended	0,357	49,32	71,30	60,313
Twofish	0,508	36,94	45,95	41,447
Sample Cipher	0,199	78,55	156,92	117,736
Cast 128	0,356	51,49	67,51	59,498
Cast 256	0,466	40,38	49,86	45,121
DES Single 8byte	0,523	37,34	42,48	39,911
DES Double 8byte	1,296	15,50	16,61	16,056
DES Double 16byte	1,288	15,52	16,83	16,173
DES Triple 8byte	1,302	15,40	16,59	15,997
DES Triple 16byte	1,288	15,52	16,80	16,163
DES Triple 24byte	1,289	15,47	16,86	16,165
DESX	0,529	35,98	43,27	39,625
Diamond II	10,315	1,99	2,04	2,016
Diamond II Lite	5,737	3,63	3,62	3,624
FROG	1,333	12,62	20,41	16,515
Mars	0,499	39,21	44,53	41,867
Misty 1	0,711	27,10	31,77	29,435
NewDES	0,726	27,21	30,21	28,711
RC2	0,858	21,91	27,08	24,492
RC4	0,466	40,85	49,06	44,953
RC5	0,266	64,51	99,14	81,824
RC6	0,385	47,59	62,48	55,034
Rijndael	0,317	57,13	76,92	67,025
Sapphire II	0,742	26,98	29,14	28,060
Skipjack	1,013	19,38	21,81	20,596

Gambar 4.13 Performa File Speed mode ECB

Dari Gambar 4.13 Pengujian *File speed* melakukan 281 iterasi, proses enkripsi Rijndael 57.13 Mb/detik dekripsi 76.92 Mb/detik dan membutuhkan waktu 0.317 detik, sedangkan Proses enkripsi Twofish 36.94 Mb/detik dekripsi 45.95 Mb/detik dan membutuhkan 0.508 detik.

4.6 PengujianKeabsahan

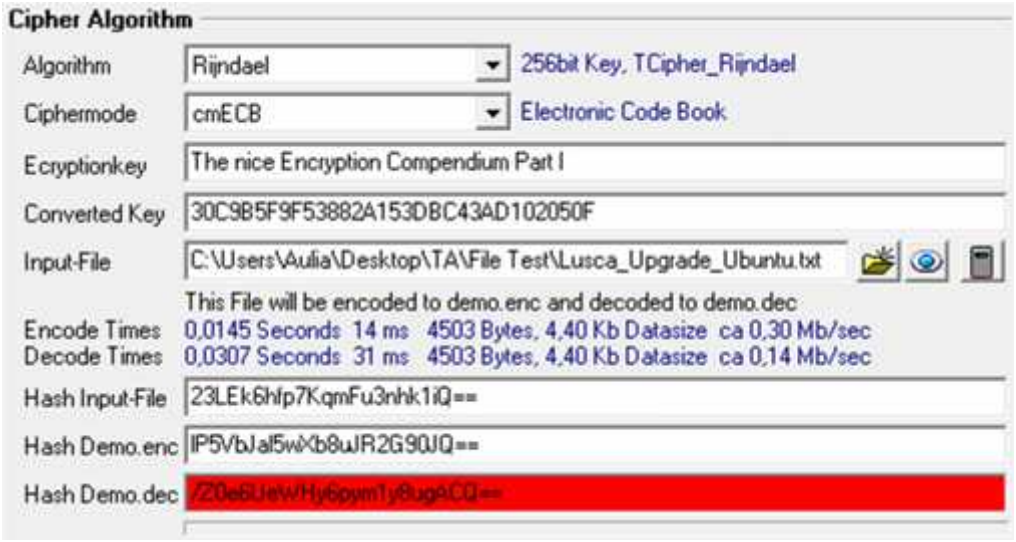
Pada tahapan ini pengujian dilakukan guna mengetahui nilai *hash* data asli (Original) dengan data yang telah di enkripsi dan dekripsidan untuk mengetahui isi *file* asli, hasil enkripsi dan hasil dekripsi.



Gambar 4. 14 Pengujian Nilai Hash

4.6.1 KeabsahanNilai Hash

Pada



Gambar 4. 14 enkripsi dan dekripsi menggunakan algoritma Rijndael mode ECB dengan *input file*“Lusca_Upgrade_Ubuntu.txt” diperoleh nilai *hash input file* dengan nilai *hash* hasil dekripsi tidak sama.

Tabel 4.6 Pengujian Keabsahan

	Algoritma	Mode Kriptografi				
		CTS	CBC	CFB	OFB	ECB
TXT	Rijndael	Sah	Sah	Sah	Sah	Tidak Sah
	Twofish	Sah	Sah	Sah	Sah	Tidak Sah
JPG	Rijndael	Sah	Sah	Sah	Sah	Tidak Sah
	Twofish	Sah	Sah	Sah	Sah	Tidak Sah
PDF	Rijndael	Sah	Sah	Sah	Sah	Tidak Sah
	Twofish	Sah	Sah	Sah	Sah	Tidak Sah
DOCX	Rijndael	Sah	Sah	Sah	Sah	Tidak Sah
	Twofish	Sah	Sah	Sah	Sah	Tidak Sah
MP3	Rijndael	Sah	Sah	Sah	Sah	Tidak Sah
	Twofish	Sah	Sah	Sah	Sah	Tidak Sah
MP4	Rijndael	Sah	Sah	Sah	Sah	Tidak Sah
	Twofish	Sah	Sah	Sah	Sah	Tidak Sah

Dari Tabel 4.6 diperoleh untuk semua tipe *file* dan mode kriptografi data *file* asli sama dengan data hasil dekrips ikecuali untuk mode ECB dari pengujian juga diperoleh nilai *hash* yang berbeda pada mode ECB untuk setiap algoritma dan *file* pengujian, berikut adalah hasil yang diperoleh dari pengujian seperti yang tertera padaTabel 4.7

Tabel 4.7 Nilai Hash

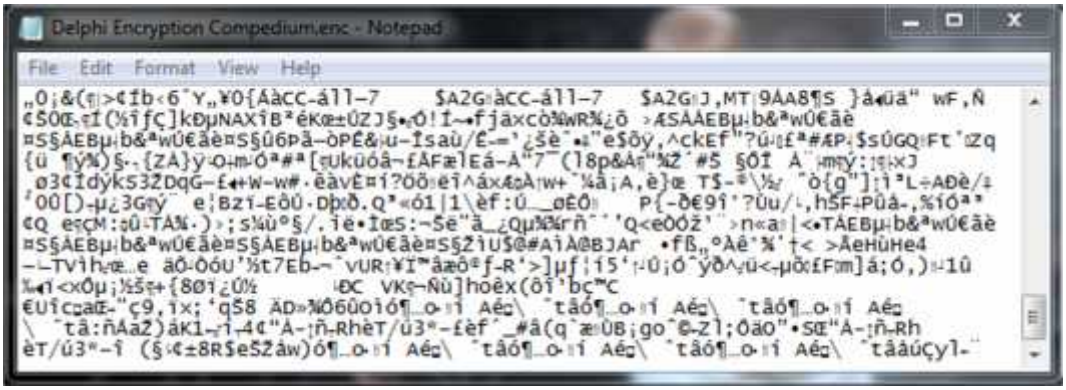
Format	Nilai Hash		
	Input	Rijndael	Twofish
TXT	23LEk6hfp7KqmFu3nhk1iQ==	/Z0e6UeWHy6pym1y8ugACQ==	G+2Xd9FvR5aus2Dzuvu3Sg==
JPG	WQpYGwbBie3MUC0mMnwabA==	ndfTo0ZMFiMnvqAkhDDWVA==	e8fkcD/AKF7ruFr5ErTWIA==
PDF	tRRdRoaR/M0dZ8RQXlA1RQ==	eviHE+03lQu2+9kmXPJzTA==	mrsjLgVbRF+wFHT6tGEEaQ==
DOCX	eM4iX7P+wgKefXUydS5zhg==	bW8/G/M3zBcWG1MHLwEBYQ==	veE/GtO+YCcyJFdF5ilUUA==
MP3	4GOs4mNA4iUA5MKlRI5r+Q==	0ooXBMNw/20CriHwyXMkVw==	YGopJqHiXmlEOFGq1PIKOQ==
MP4	ax9AZi82LQA1xAGGYQO0oQ==	luEy/tN1v8HiJThp21DpLA==	veHtU6NFb9K67GlZ+q5LFg==

4.6.2 Keabsahan Isi File

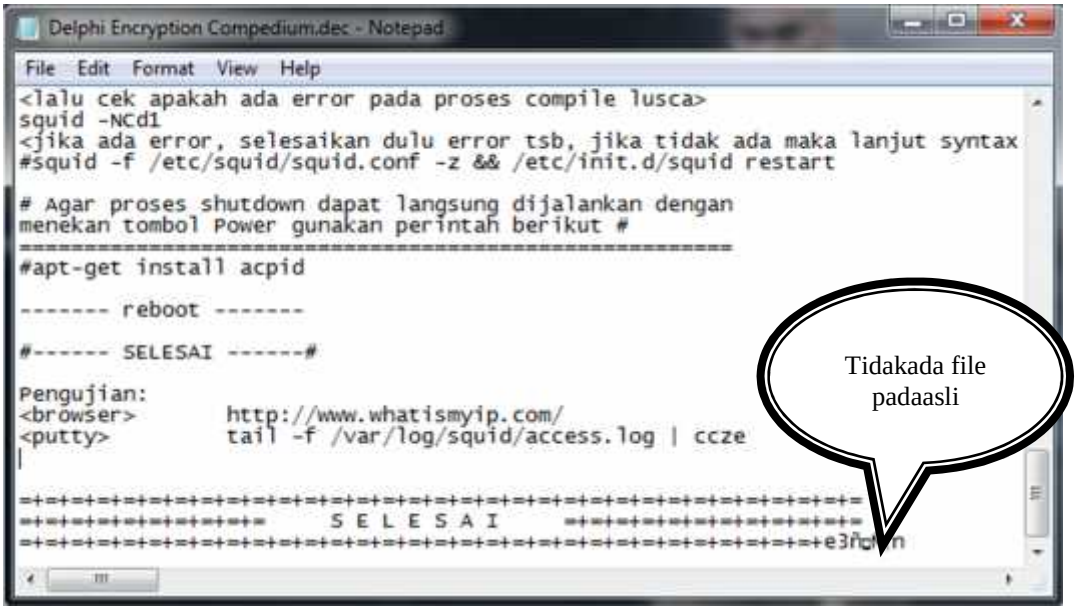
Pada Gambar 4.15 ditampilkan sebagian isi *file* di bagian akhir, ini diperlukan untuk membandingkan isi *file* tersebut dengan isi *file* hasil dekripsi pada Gambar 4.17, pada pengujian ini diperoleh isi *file* sama atau tidak ada perubahan dengan hasil dekripsi kecuali pada bagian akhir ”=+=” menjadi “e3ñÑn”

Gambar 4.15 Data Asli

Pada gambar Gambar 4. 16 menunjukan bahwateknik *diffusion* dan *confusion* berhasil diterapkan implementasi algoritma pada Delphi Encryption Compedium karena isi *file* udah tidak kemiripan pada *file* asli juga kemunculan karakter pada hasil dekripsi pada *file* hasil dekripsi juga sudah tidak dapat ditentukan pola karakter.



Gambar 4. 16 Data Hasil Enkripsi



Gambar 4.17 Data Hasil Dekripsi

Pada gambar diatas diketahui ada penambahan karakter “e3ñÑn” yang sebelumnya tidak ada pada *file* asli.

4.6.3 Ukuran File

Dari pengujian yang dilakukan tidak ditemukan ada perubahan ukuran file dengan kata lain ukuran *file* asli sama dengan ukuran *file* hasil enkripsi juga sama dengan ukuran *file* hasil dekripsi.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan studi, analisa, dan pengujian yang dilakukan, penulis menyimpulkan:

1. Kedua algortima menggunakan fungsi *Add* , *substract* atau operasi logika. Pada keduanya juga dijumpai penggnaan tabel *lookup*, operasi ini sangat bergantung pada *resouce memory*.
2. Arsitektur, struktur dan komponen *Rijndael* cenderung lebih fleksibel, effisien dan memiliki performa yang sangat bagus.
3. Arsitektur, struktur dan komponen *Twofish* cenderung memiliki tingkat *security* yang tinggi, kombinasi desain arsitektur komponen MDS, *Rotate byte*, *q-permutation* PHT dan *S-box* yang dibangun dari kombinasi 3 buah *q-permutation* (4 *byte*) menjadi kunci kekuatan yang dapat membuat para kriptanalis harus kerja keras untuk memecah *cipher*.
4. *Bit-bit* yang dihasilkan pada setiap *round*-nya (*rijndael* dan *twofish*) dibangun berdasarkan dan masih berhubungan (berkorelasi) dengan *bit-bit* sebelumnya.
5. Penjadwalan dan pengembangan kunci pada algortima rijndael dibangun terpisah jadi untuk setiap roundnya *subkey* sudah dapat di ditentukan, sedangkan untuk rijndael *bit-bit* key (*subkey* 9 – 19) harus dikembangkan bersama dengan *bit-bit* data .

5.1 Saran

1. Bagi praktisi pengembang aplikasi enkripsi sebaiknya mempertimbangkan mamfaat kedua algoritma.
2. Implementasi pada *hardware* sebaiknya harus diperhatikan dukungan-dukungan set instruksi yang dapat didukung oleh kedua algoritma.
3. Untuk mendukung performa yang bagus dan dukungan blok data 128 *bit* sebaiknya menggunakan algoritma Rijndael.

4. Untuk mendukung tingkat keamanan data sebaiknya gunakan algoritma Twofish.

DAFTAR PUSTAKA

Canright D., *A Very Compact Rijndael S-Box*. New York: Storming Media, 2004.

Daemen Joan dan Rijment Vincent. *The Design of Rijndael*. Berlin: Springer, 2002.

Goots Nick, Izotov Boris, Moldovyan Alexander, Moldovyan Nick. *Modern Cryptography: Protect Your Data with Fast Block Ciphers*. Wayne : LLC A-List, 2003.

Menez A., Van Oorschot P., dan Vanstone S.,. *Handbook of Applied Cryptography*. CRC Press, 1996.

Newton, David E., *Encyclopedia Of Cryptology* .California : ABC-Clio, Inc., 1997.

Rotman, Joseph J., *Advance Modern Algebra.* New Jersey : Prantice Hall, 2003.

Shannon C.E. *A Mathematical Teory Of Communication*. Vol 27, Halaman 379-423, dan 623-656. Dicetak ulang oleh The Bell System Technical Journal, Oktober 1948.